UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di laurea in Ingegneria dell'Informazione

Progettazione di un'interfaccia di annotazione lessicale per un programma di Quantitative Narrative Analysis

LAUREANDO

Luca Schiappadini

RELATORE

Cinzia Pizzi

CORRELATORI

Roberto Franzosi

Fabio Cunial

ANNO ACCADEMICO 2012/2013

Abstract

Nelle scienze sociali l'analisi di documenti (testi, articoli di giornale, etc.) relativi agli eventi storici oggetto di studio è un passaggio di fondamentale importanza per capire le cause che hanno scatenato tali eventi e le loro conseguenze. In tempi recenti c'è stato un notevole sviluppo di software dedicato che permette di assistere i sociologi nella loro ricerca. In questo contesto, questa tesi ha come obiettivo il miglioramento delle prestazioni della procedura d'inserimento dei dati in un programma che compie Quantitative Narrative Analysis, una particolare tecnica di analisi dei testi che, come suggerisce il nome, si focalizza sull'estrazione di informazione quantitativa, e quindi oggettiva, contenuta nei testi esaminati. Si è proceduto in due diverse direzioni. Dal punto di vista applicativo si è sviluppata una nuova interfaccia grafica per la procedura d'immissione dei contenuti delle fonti documentali in sostituzione alla precedente. Come ulteriore contributo è stato effettuato uno studio riguardante la possibilità di applicare tecniche di Information Extraction ai documenti sottoposti all'analisi. Questo, in prospettiva, permetterebbe di eseguire l'acquisizione dei dati in modo automatico o al più semiautomatico, consentendo di concentrarsi maggiormente sulla loro analisi, vero obiettivo di chi si occupa di studi nell'ambito delle scienze sociali.

Indice

1 Introduzione

Uno dei problemi fondamentali nell'ambito delle scienze sociali è l'analisi delle relazioni esistenti tra i vari eventi che costituiscono il movimento storico d'interesse del ricercatore. Questo tipo di analisi è compiuto al giorno d'oggi principalmente attraverso l'utilizzo di strumenti informatici, in quanto solo l'elevata capacità di calcolo offerta dagli elaboratori permette di condurre in tempi brevi indagini raffinate su grandi moli di dati. Diversi sono i software, commerciali e non, sviluppati con l'intento di aiutare i sociologi a condurre le loro ricerche. Tra questi PC-ACE (Program for Computer – Assisted Coding of Events) è stato creato con l'obiettivo di compiere Quantitative Narrative Analysis [3], una particolare tecnica di ricerca quantitativa eseguita a partire da informazioni estratte da testi narrativi, ad esempio gli articoli di giornale. PC-ACE si distingue dagli altri software per l'analisi nelle scienze sociali per le numerose funzionalità che mette a disposizione per l'analisi quantitativa dei documenti.

Il lavoro svolto in questa tesi è consistito nell'implementazione di un'interfaccia grafica che permettesse l'immissione dei dati nel database di PC-ACE in modo più semplice e veloce rispetto a quanto si poteva fare in precedenza, basandosi anche su un confronto con alcuni programmi commerciali usati nell'ambito dell'analisi testuale nelle scienze sociali [4]. È stato inoltre eseguito uno studio delle tecniche di Information Extraction [10] per argomentare la possibilità di compiere l'acquisizione dei dati in modo automatizzato attraverso di esse. Infatti, in ogni indagine compiuta tramite l'analisi dei testi questa fase rappresenta quella più avida in termini di tempo e un suo svolgimento senza intervento di un operatore umano ridurrebbe di molto i tempi richiesti per completare una ricerca.

La tesi è strutturata nel seguente modo: nel Capitolo 2 viene introdotto il problema dell'analisi testuale nelle scienze sociali, nel Capitolo 3 viene descritta più in dettaglio la struttura di PC-ACE, nei Capitoli 4 e 5 viene descritta l'implementazione della nuova interfaccia di inserimento dei dati, dividendo l'esposizione dell'interfacciamento con PC-ACE dai nuovi controlli creati per l'interfaccia, mentre nel Capitolo 6 viene trattata la possibilità di automatizzare l'inserimento dei dati sfruttando tecniche di Information Extraction.

2 Analisi testuale nelle scienze sociali

Durante lo svolgimento di una ricerca nell'ambito delle scienze sociali si hanno a disposizione varie tecniche per procedere alla raccolta dei dati. Tali tecniche sono classificate di norma in base alla loro collocazione sull'asse qualitativo-quantitativo. Si dicono infatti qualitative se si concentrano sulla comprensione dei fenomeni sociali tramite l'osservazione diretta dei fatti o l'analisi dei testi, concentrandosi dunque sulla qualità dei dati e perdendo spesso di generalità. Si dicono invece quantitative se si approcciano agli eventi sociali basandosi su un approccio statistico, concentrandosi dunque sulla quantità dei dati mirando a definire delle considerazioni di carattere generale. L'insieme delle metodologie di acquisizione dei dati è dunque molto eterogeneo, spaziando, ad esempio, dall'analisi dei testi ad indagini storiche o di archivio ad interviste e sondaggi. In seguito sono descritte alcune caratteristiche delle tecniche di analisi testuale più comuni per procedere poi ad una introduzione alla Quantitative Narrative Analysis, la tecnica di analisi applicata da PC-ACE, il software utilizzato in questa tesi.

2.1 Tecniche di analisi testuale

L'analisi testuale, detta anche *content analysis* [1], consente di analizzare raccolte di testi, anche molto vaste, di particolare interesse nell'ambito di una specifica ricerca. Essa si può definire come una tecnica di compressione sistematica che assegna una o più parole di un documento ad alcune categorie basandosi su esplicite regole di codifica [2]. La sua diffusione nel campo delle scienze sociali è coincisa con lo sviluppo dell'informatica e con l'aumento delle capacità di calcolo degli elaboratori avvenuta dall'inizio degli anni '90, durante i quali è stata impressa una decisa accelerazione in questo settore grazie al crescente sviluppo di software dedicati. Anche se classificate di norma come qualitative queste tecniche consentono di intraprendere anche un approccio all'analisi di tipo quantitativo.

L'analisi testuale si basa sull'assunto secondo il quale le parole, le frasi, i temi o i concetti che compaiono in numero maggiore all'interno di un gruppo di testi sono gli aspetti più importanti ai fini della comunicazione eseguita da quell'insieme di documenti. Un approccio quantitativo, quindi, avrà come prima fase dell'analisi vera e propria il conteggio della frequenza delle varie parole nei testi in esame, con la possibilità di aggregare parole diverse in un unico concetto ai fini dell'analisi.

Per condurre questo tipo di analisi il testo deve essere codificato in diverse categorie, specificate all'inizio della ricerca in base agli obiettivi che s'intendono perseguire. Queste

categorie possono essere associate ad ogni livello del discorso, sia esso parola, frase o tematica. La codificazione rappresenta il passo fondamentale dell'intero processo, in quanto solo tramite questa riduzione selettiva in categorie significative si può condurre un esame del messaggio contenuto nel testo. Lo schema delle categorie ottenuto dall'operazione di estrazione delle informazioni rappresenta comunque uno dei punti deboli di questo processo, dipendendo nella struttura dalla ricerca specifica e nell'attribuzione delle parti del testo alle varie categorie dalla persona che compie l'operazione di codificazione. L'analisi vera e propria avviene comunque su questi elementi e consiste essenzialmente nell'analisi della frequenza con cui compaiono nel testo e delle relazioni presenti tra di essi.

2.2 Quantitative Narrative Analysis

La Quantitative Narrative Analysis [3] è un metodo di ricerca quantitativa che mira ad estrarre informazioni quantitative da testi narrativi. L'analisi avviene attraverso l'assegnazione di parole o locuzioni tratte dai documenti in esame (generalmente articoli di giornali contemporanei agli eventi analizzati) agli oggetti che costituiscono un prefissato schema di codifica, la cosiddetta *Story Grammar*, che cattura quindi gli elementi base della storia e come sono relazionati. Attraverso l'analisi della frequenza con cui le varie parole compaiono nei diversi oggetti della grammatica e della loro correlazione si può dunque eseguire un'analisi statistica di un testo narrativo.

Questa tecnica di analisi si differenzia dalle altre già esistenti nel campo delle scienze sociali poiché le sue categorie di codifica non dipendono né dal progetto specifico né dal ricercatore, ma si basano invece su strutture invarianti dei testi narrativi, ossia la sequenzialità dei fatti descritti e la possibilità quindi di ridurre ogni storia descritta in questi testi ad una successione di triplette del tipo Soggetto-Azione-Oggetto. Ciascuna di queste triplette descrive una singola azione: solo raramente sono di natura descrittiva o valutativa [3].

I passi fondamentali per compiere la Quantitative Narrative Analysis di un gruppo documenti sono:

- Definizione di una Story Grammar;
- Estrazione dei dati dai vari testi;
- Analisi statistica.

Poiché il secondo ed il terzo passo verranno descritti in dettaglio nei capitoli successivi ci si limita, in questo contesto, ad approfondire la definizione di Story Grammar.

2.2.1 Story Grammar

La struttura Soggetto-Azione-Oggetto identifica una Story Grammar, che per un dato evento può essere arbitrariamente complessa. Infatti ai tre elementi canonici sopra descritti possono essere aggiunti un numeroso gruppo di modificatori che ne specificano i dettagli: ad esempio all'Azione possono essere aggiunti dei modificatori per indicarne il tempo, il luogo, la ragione etc. Si ottiene in questo modo un ampio gruppo di oggetti relazionati tra loro che vanno a formare una struttura ad albero, la cui creazione è espressa in modo rigoroso tramite delle regole di riscrittura (o produzioni). Questa sintassi è espressa in forma di Backus-Naur: una Story Grammar costituisce, quindi, una grammatica libera dal contesto. Tramite queste specifiche la tripletta Soggetto-Azione-Oggetto si esprime formalmente come:

<tripletta semantica> → {<soggetto>} {<azione>} [{<oggetto>}]

Nello stesso modo si può scomporre ogni elemento della grammatica, ad esempio:

{<soggetto>} → {<attore>} [{<caratteristiche>}]
<attore> → uomo | donna | folla | sconosciuto | …
<caratteristiche> → [<nome>] [<tipo>] [<numero>] [{<organizzazione>}] …

in cui le frecce definiscono le produzioni da oggetto padre sulla sinistra ad oggetti figli sulla destra. Per quanto riguarda la notazione, le parentesi quadre indicano che un elemento è opzionale, le parentesi graffe che un elemento può essere inserito più volte e le parentesi angolari che per quell'elemento esiste una regola di riscrittura, mentre l'assenza di queste parentesi indica che un elemento è terminale ossia che è una parola o una locuzione proveniente dal testo da cui la Story Grammar è stata compilata, in analogia dunque con le regole di definizione della forma di Backus-Naur.

Esistono due tipi di oggetti della grammatica: i *simplex object*, la cui riscrittura avviene inserendo un elemento terminale, e i *complex object*, la cui riscrittura avviene tramite altri simplex e/o complex. Le triplette semantiche, inoltre, possono non costituire il livello più alto di aggregazione delle informazioni nelle Story Grammar: è possibile infatti unire le triplette per formare oggetti più grandi come gli eventi e aggregare anche questi ultimi in storie o macroeventi, in modo che tutte le informazioni riguardanti un singolo fatto siano memorizzate in un'unica struttura.

L'estrazione delle informazioni dai testi tramite schemi di questo tipo comporta diversi vantaggi rispetto ai metodi di analisi testuale usati normalmente nelle scienze sociali. Innanzitutto i legami gerarchici e relazionali tra i vari oggetti della grammatica sono stabiliti esplicitamente attraverso le regole di riscrittura. Inoltre al termine della codifica l'output mantiene sia una forma facilmente comprensibile per un operatore umano sia la maggior parte delle informazioni presenti nel testo di partenza, non essendo l'acquisizione dei dati concentrata su un aspetto specifico o basata sull'aggregazione di voci diverse.

Come principale svantaggio rispetto le altre tecniche si nota come la Quantitative Narrative Analysis non sia adatta all'elaborazione di grandi contenuti descrittivi. Questo tipo di analisi, infatti, è adatto solo a testi "densi" di azioni poiché le frasi descrittive non possono essere ricondotte facilmente alla struttura Soggetto-Azione-Oggetto.

3 Descrizione di PC-ACE

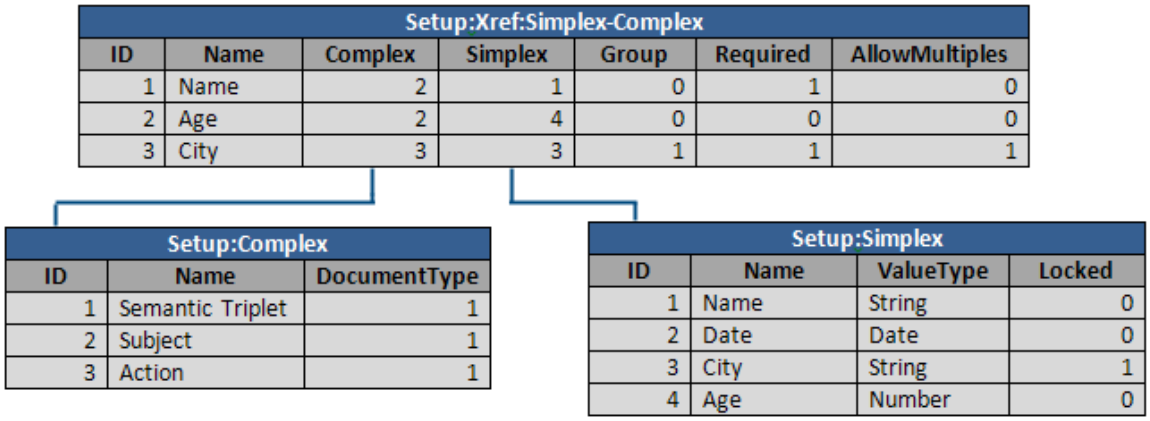
L'approccio alternativo che comporta la Quantitative Narrative Analysis rispetto agli altri metodi di analisi testuale e la complessità della Story Grammar appena descritta non consente di applicare questa tecnica di analisi utilizzando software già esistenti specifici per l'analisi testuale in ambito sociologico [4]. Questo ha portato allo sviluppo di PC-ACE (Program for Computer – Assisted Coding of Events), un programma open-source operante sulla piattaforma Windows che rende possibile una sua implementazione pratica. In questo capitolo sarà illustrata brevemente la sua struttura interna, mettendo in risalto gli elementi importanti per la comprensione del funzionamento della nuova interfaccia d'inserimento, e saranno presentati alcuni esempi del suo utilizzo.

3.1 Struttura interna

La versione attuale di PC-ACE rappresenta la sua seconda implementazione, dopo che la prima, realizzata in Turbo Pascal per MS DOS alla fine degli anni Ottanta, fu abbandonata in quanto divenuta, nel frattempo, obsoleta. Nel 2002 si eseguì, infatti, una completa riscrittura del codice adattandolo all'interfaccia grafica di Windows e appoggiandosi a Microsoft Access. La versione corrente (828) per Microsoft Access 2010 è completamente scritta in VBA (Visual Basic for Application), esclusi due moduli specifici, per un totale di circa 80.000 righe di codice divise in 127 moduli che operano su 24 tabelle principali e un'altra decina di supporto.

La scelta di sviluppare la nuova versione di PC-ACE appoggiandosi ad un applicativo esistente, presa probabilmente per ridurre i tempi di sviluppo rispetto a quelli richiesti da una creazione da zero del programma, si presenta ora come il problema più grande di PC-ACE, in quanto ne limita sia le prestazioni che la portabilità che l'espansibilità. Un'altra barriera alla manutenzione e all'espansione del software è data dal fatto che la quasi totalità del codice sviluppato nel periodo 2002-2005, costituente la parte del programma che gestisce il database e il suo interfacciamento con l'esterno, non è documentata ed è quindi di difficile comprensione.

Il Database Management System utilizzato in PC-ACE è basato sul modello relazionale: i dati sono quindi salvati in numerose tabelle diverse collegate tra loro tramite dei campi in comune. Ad esempio un elemento terminale come il nome del soggetto di una tripletta semantica è salvato in una tabella, la sua relazione con l'oggetto padre, il soggetto, in un'altra e un elemento non terminale come un soggetto in un'altra ancora, con i collegamenti



Setup:Xref:Simplex-Complex

ID	Name	Complex	Simplex	Group	Required	AllowMultiples
1	Name	2	1	0	1	0
2	Age	2	4	0	0	0
3	City	3	3	1	1	1

Setup:Complex

ID	Name	DocumentType
1	Semantic Triplet	1
2	Subject	1
3	Action	1

Setup:Simplex

ID	Name	ValueType	Locked
1	Name	String	0
2	Date	Date	0
3	City	String	1
4	Age	Number	0

Figura 1 Esempio delle relazioni esistenti tra le tabelle di PC-ACE, in particolare tra tre di quelle in cui è definita una Story Grammar.

realizzati tramite dei set di ID univoche per tabella (vedere la Figura 1). Quest'architettura comporta un ulteriore appesantimento delle prestazioni: basti pensare che per caricare completamente una Story Grammar possono essere necessari anche diverse migliaia di accessi alle tabelle. Queste prestazioni sono però compensate dalla robustezza e della compattezza che questo approccio comporta.

Il reperimento dei dati all'interno del database da parte dell'utente finale avviene attraverso query scritte in linguaggio SQL, supportato in modo nativo da Microsoft Access, il cui utilizzo è filtrato opportunamente tramite un'interfaccia grafica: un generico ricercatore non sarebbe in grado di scrivere le query per estrarre i dati, non essendogli probabilmente nota l'organizzazione interna delle tabelle.

Si riportano di seguito alcune soluzioni adottate nell'architettura che permettono una maggiore flessibilità nell'applicazione della teoria della Quantitative Narrative Analysis.

3.1.1 Organizzazione di una Story Grammar

L'organizzazione della Story Grammar usata nel progetto di ricerca non dipende dalla struttura delle tabelle in cui sono memorizzati i dati, ma è definita in quattro tabelle dedicate e può dunque essere modificata, se necessario, nel corso di un progetto e non solo al suo avviamento. Simplex object, complex object e le relazioni esistenti rispettivamente tra simplex e complex e tra complex e complex sono quindi definiti tutti in tabelle diverse. In questo modo un qualsiasi oggetto può essere copiato e definito come figlio anche di più oggetti, non dovendo quindi ridefinire più volte uno stesso oggetto per usarlo in parti diverse della grammatica. È possibile anche definire degli *alias* di un oggetto, ossia istanze con nomi diversi che si basano sullo stesso oggetto e avranno quindi la stessa ID: ad esempio l'oggetto <città> può avere come alias <luogo di residenza>. La suddivisione tra simplex e complex, ovvero tra nodi foglia e nodi interni della Story Grammar, è presente anche al livello più basso dell'architettura e li porta a non condividere lo stesso set di ID identificative ma ad averne due paralleli.

3.1.2 Organizzazione dei documenti

I documenti su cui viene compiuta l'analisi, inoltre, sono trattati come oggetti a sé stanti, simili a delle Story Grammar semplificate, aventi quindi le loro proprie regole di riscrittura per memorizzare vari dettagli come titolo, data e simili. Anche la loro definizione è simile a quella della grammatica: è salvata infatti in due tabelle, una dedicata alla definizione dei tipi di documento e una dei simplex object a loro collegati. In questo modo ogni oggetto delle Story Grammar vere e proprie può contenere uno (nel caso dei simplex object) o più (nel caso dei complex object) riferimenti ai documenti dai quali sono stati tratti, rendendo possibili ricerche in base ai testi di origine.

3.1.3 Dizionario dei simboli terminali

Sono presenti infine alcune tabelle particolari che hanno la funzione di dizionario per gli elementi terminali della Story Grammar, in cui ad ogni valore è assegnato una ID univoca e l'ID dell'elemento in cui sono state inserite: in questo modo si evita di aggiungere più volte lo stesso dato per lo stesso oggetto terminale. Per facilitare l'analisi statistica dei dati questi valori devono però essere aggregati in categorie più ampie poiché molti di essi appaiono con frequenza molto bassa o addirittura unitaria: in un progetto tipico come quello sui linciaggi avvenuti in Georgia illustrato in seguito si arrivano ad avere anche più di 5.000 triplette con, ad esempio, anche 1.000 valori diversi per le azioni [5]. Per questo motivo di fianco a molti elementi è presente un campo per inserire il loro codice aggregato: in questo modo quindi azioni come "scappò", "percorse", "seguì" e "arrivò" sono tutte classificate come "movimento". L'assegnazione di questi aggregatori deve comunque essere compiuta manualmente durante l'inserimento dei dati o al completamento di questa procedura, filtrando in questo caso i diversi valori e assegnando loro una classificazione. In un'ottica di un'espansione del programma per ottenere una maggiore automazione delle operazioni sarebbe utile implementare un'assegnazione automatica dei codici aggregati, ad esempio attraverso un dizionario precompilato aggiornato via via che parole non presenti sono inserite.

3.2 Funzionalità

PC-ACE consente di compiere numerose operazioni sui dati memorizzati all'interno del suo database, anche se la loro elaborazione statistica e le funzioni di visualizzazione più avanzate sono disponibili solo sfruttando programmi esterni. Si può quindi vedere PC-ACE come un sofisticato sistema di archiviazione che ha l'obiettivo principale di assistere il ricercatore in una serie di operazioni [6]:

- Creare strutture consistenti.
- Velocizzare le operazioni d'inserimento e di validazione delle informazioni.
- Rendere l'estrazione delle informazioni il più flessibile possibile.

Con i dati ottenuti svolgendo una query nel database è possibile eseguire diversi tipi di elaborazione. Ad esempio, sfruttando la flessibilità dell'istruzione SQL `count`, è possibile determinare la distribuzione di frequenza di un oggetto in relazione con altri per poi esportarla da PC-ACE ed eseguire l'elaborazione statistica dei dati usando dei programmi specifici. Con queste applicazioni è possibile eseguire diversi tipi di analisi importanti nelle scienze sociali come l'analisi delle sequenze (che sfrutta la sequenzialità delle triplette semantiche in eventi simili per estrarre dei *pattern* e analizzarli), della regressione (in cui viene creato un modello che stima il valore di una variabile dipendente in funzione di un qualsiasi numero di variabili

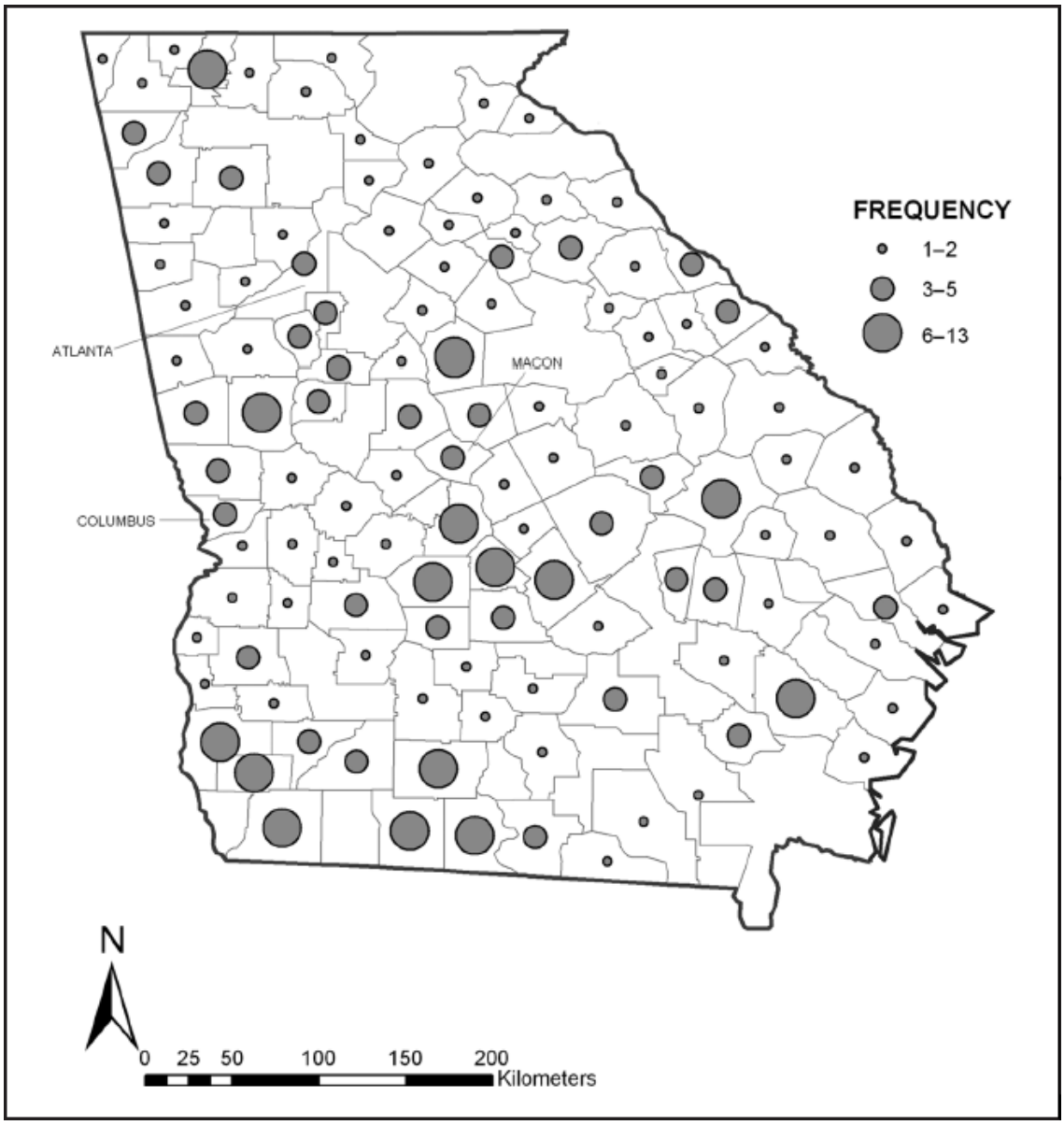


Figura 2 Distribuzione degli eventi di linciaggio nelle contee della Georgia (1875 - 1930), tratto da [5].

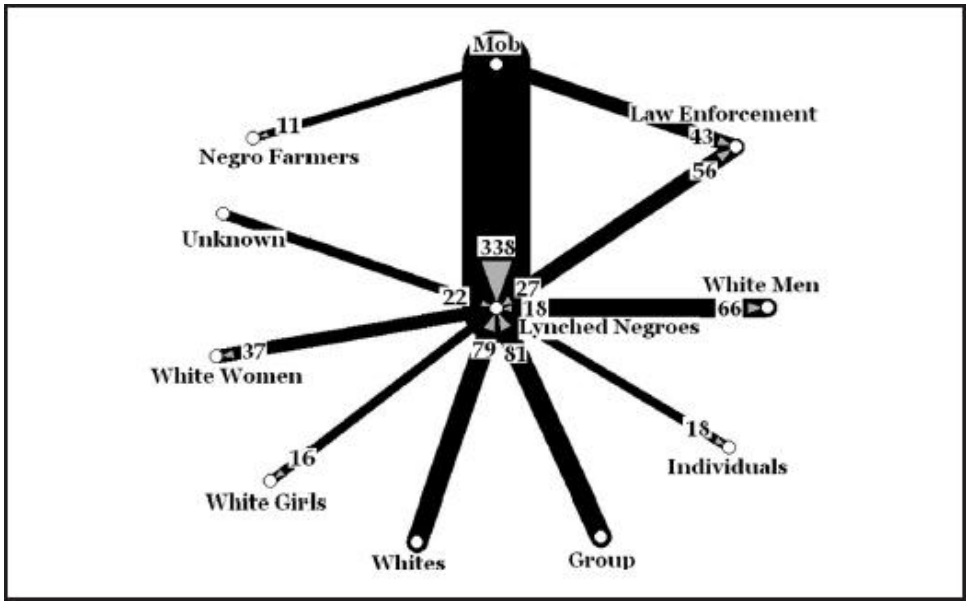


Figura 3 Network della violenza contro le persone (Georgia, 1875 - 1930), tratto da [5].

indipendenti) e delle corrispondenze (che ha l'obiettivo di individuare variabili sottostanti al livello dei dati che riassumono le relazioni di dipendenza delle variabili originarie). È possibile inoltre appoggiarsi a programmi esterni anche per fornire funzionalità avanzate di visualizzazione. Ad esempio è possibile esportare i risultati delle query come elementi grafici in programmi GIS (Geographic Information System, vedere la Figura 2), per evidenziare la loro distribuzione geografica oppure sotto forma di grafi in programmi come Gephi (vedere la Figura 3), per evidenziare i collegamenti tra i vari oggetti.

La versione attuale di PC-ACE presenta alcune limitazioni. Innanzitutto, come si è visto, non si ha la possibilità di compiere un'elaborazione statistica approfondita dei dati all'interno del programma, richiedendo quindi l'installazione di vari applicativi esterni per poterla eseguire. PC-ACE non è inoltre dotato né di una funzione di estrazione automatica delle informazioni dai testi, né di una funzione di acquisizione dei dati automatizzata a partire da un documento opportunamente formattato: ogni elemento delle Story Grammar deve essere compilato a mano dal ricercatore.

Tutto ciò non toglie che PC-ACE sia un programma dotato di una grande flessibilità che è stata opportunamente sfruttata in numerosi progetti da diversi ricercatori americani ed europei. Un progetto ideale per grandezza e risultati da cui trarre degli esempi delle sue potenzialità riguarda i linciaggi avvenuti nello stato della Georgia tra il 1875 e il 1930 [5] in cui sono state compilate più di 7000 triplette semantiche a partire da più di 1000 documenti. Inoltre dalla sua presentazione sono state tratte le immagini portate come esempio in questa sezione e sul suo database sono stati portati a termine tutti i test realizzati per la scrittura di questa tesi.

4 L’interfaccia per l’inserimento dei dati

Progetti di studio come quello riguardante i linciaggi avvenuti in Georgia si basano sull’acquisizione di dati da qualche migliaio di documenti. In contesti come questo l’impossibilità di acquisire automaticamente le informazioni dai documenti, e la necessità quindi di inserire a mano i dati, può rappresentare una limitazione all’utilizzo di PC-ACE. Questa tesi propone una nuova interfaccia per l’inserimento dei dati in PC-ACE, permettendo la riduzione del tempo necessario a completare l'estrazione delle informazioni da un documento. Nel medio-lungo termine l’obiettivo è tuttavia quello di automatizzare, parzialmente o totalmente, il processo di acquisizione dati. Nei prossimi due capitoli si descriveranno le caratteristiche della nuova interfaccia d’inserimento dei dati mentre un possibile approccio all’automatizzazione dell’acquisizione sarà discusso nel Capitolo 6.

4.1 Limitazioni della versione precedente

L’inserimento dei dati in PC-ACE attraverso la maschera attuale (vedere la Figura 4)

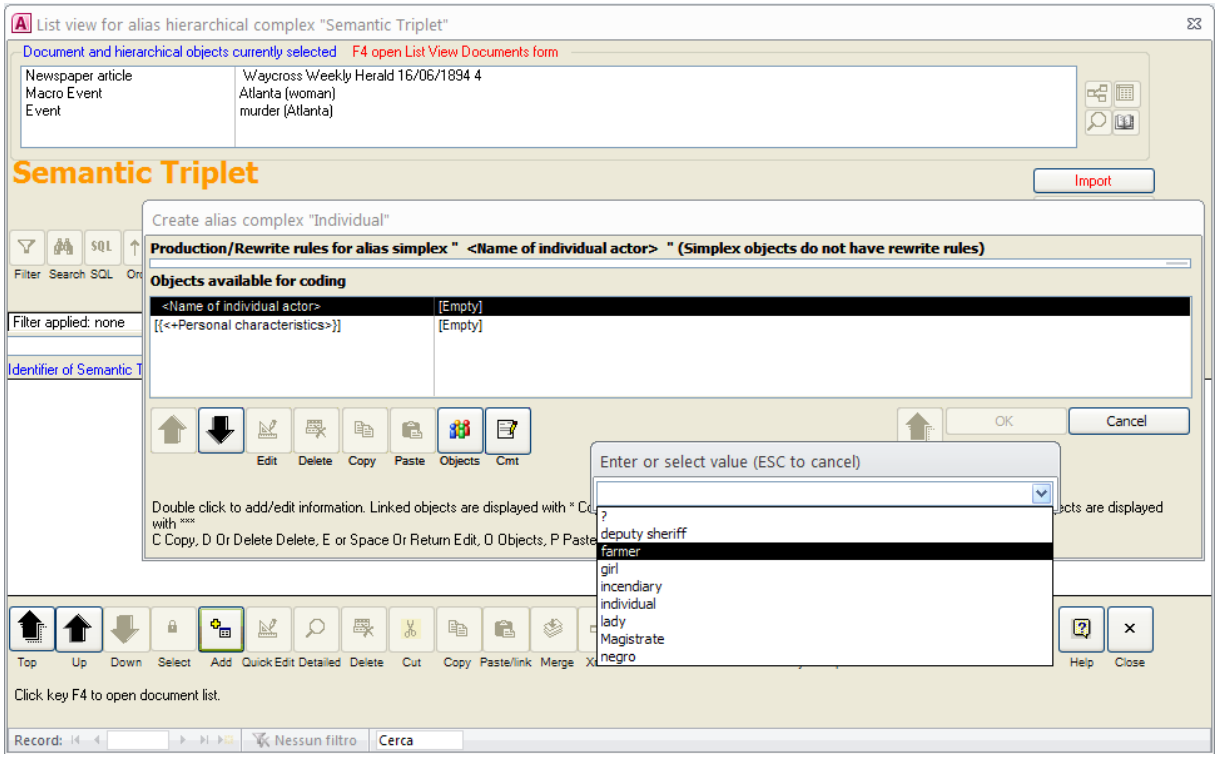


Figura 4 Maschera e finestre di dialogo per l'inserimento dei dati di PC-ACE; si nota, partendo dal fondo, la maschera che visualizza la tripletta semantica di partenza, la sottomaschera per la creazione di un complex e la finestra di dialogo per l’inserimento di un valore terminale.

può essere compiuta solo lentamente e senza sfruttare tutte le potenzialità offerte dal programma. Si consideri infatti che in media da ogni testo sono ricavate tra le 5 e le 8 triplette semantiche e che un buon operatore riesce a completare l'estrazione dei dati da un documento in un intervallo compreso tra i 20 e i 30 minuti [6]. Oltre che dalla lunghezza caratteristica di questo genere di operazione, questi tempi sono determinati anche dal fatto che la stessa interfaccia sia usata sia per l'aggiunta di dati nel database che per la loro revisione, e il design dell'interfaccia risulta sbilanciato verso la seconda operazione, complicando indesideratamente la prima.

Il primo limite che si nota procedendo alla compilazione di una Story Grammar tramite questo form è l'impossibilità di visualizzare direttamente all'interno della maschera d'inserimento il documento su cui si sta lavorando. Anche se al suo inserimento nel database è stato specificato un file di origine, si ha infatti la possibilità di aprirlo solamente in un'altra finestra, utilizzando il programma associato a quel formato dal sistema operativo.

Un altro rallentamento delle operazioni d'inserimento è dovuto al fatto che tutti gli spostamenti tra i nodi dell'albero che costituisce la Story Grammar su cui si sta lavorando possono avvenire solo in modo sequenziale, passando cioè da padre a figlio o viceversa. L'unica opportunità di navigazione veloce, infatti, è rappresentata dalla possibilità di tornare al primo livello in cui non si sono completati oggetti marcati come obbligatori. Ogni movimento nella struttura provoca inoltre l'aggiornamento della sottomaschera in cui è visualizzato l'oggetto che si sta completando, che viene in questo modo sostituito dai suoi figli, perdendo di vista quindi l'insieme dei dati inseriti fino a quel momento.

Infine, come si nota sempre dalla Figura 3, l'inserimento dei valori negli elementi terminali avviene tramite o la scelta dei possibili valori da una ComboBox o l'inserimento manuale degli stessi in una TextBox contenute in una finestra di dialogo a parte.

Le modifiche da apportare per semplificare e velocizzare la procedura d'inserimento dei dati appaiono dunque chiare e in linea con quanto emerso dal confronto di PC-ACE con altri software commerciali di analisi e manipolazione dei dati correntemente usati in campo sociologico [4]. Bisogna infatti provvedere alla visualizzazione del documento da cui si stanno estraendo i dati direttamente all'interno di PC-ACE e a visualizzare la Story Grammar che si sta creando in modo integrale, sfruttando quindi in modo completo la sua struttura ad albero, che è quasi ignorata nel design attuale.

4.2 Descrizione della nuova maschera di inserimento

Le osservazioni precedenti hanno portato alla creazione della nuova maschera d'inserimento dei dati che si può osservare in Figura 5. Per facilitare l'interazione con un utente non esperto nell'uso del programma si è scelto di utilizzare un design semplice, dividendo quindi il form in 3 zone:

1) La zona superiore, che occupa la maggior parte dell'area disponibile, contiene i due controlli principali: DocumentViewer nella parte sinistra, che consente la visualizzazione del documento su cui si sta operando, e AdvancedTree nella parte destra, nel quale è mostrata la Story Grammar su cui si sta operando ed attraverso il quale è quindi possibile inserire i dati, compilando le TextBox presenti di fianco ai simplex object. Per una descrizione in dettaglio del funzionamento dei due controlli vedere il Capitolo 5.
2) La zona intermedia, che contiene i pulsanti di comando della maschera.
3) La zona inferiore, in cui vengono visualizzati suggerimenti riguardati i pulsanti o i controlli.

Le ultime due aree sono presenti in tutte le altre maschere di PC-ACE e sono quindi state mantenute per uniformità con il layout del programma, insieme allo stile dei pulsanti e degli

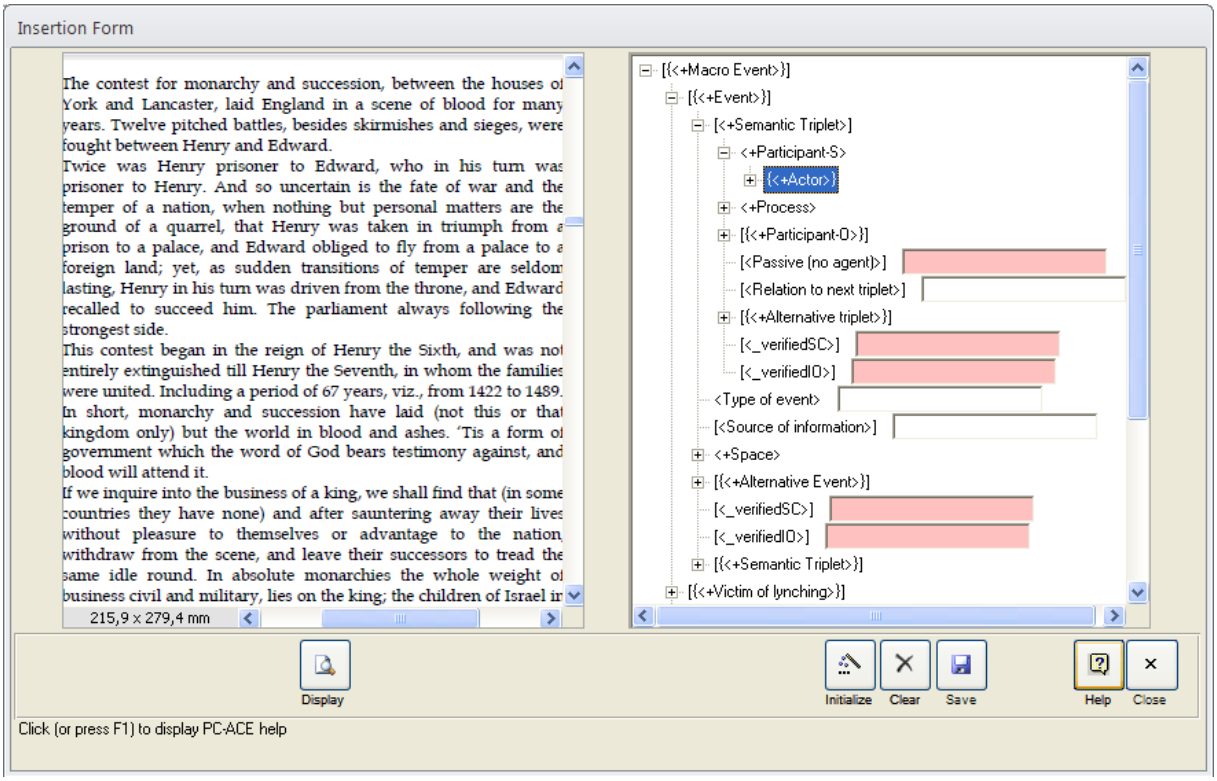


Figura 5 La nuova maschera per l'inserimento dei dati in PC-ACE.

altri elementi grafici e con i tasti Help e Close, che, rispettivamente, aprono la guida del programma e chiudono il form corrente.

Le operazioni d'inserimento dei dati si dimostrano ora molto più semplici e veloci rispetto a quanto avveniva in precedenza. Infatti, dopo aver scelto il documento da cui si vogliono estrarre i dati (operazione che avviene tramite la maschera di selezione del documento attivo già presente in PC-ACE), basta premere il pulsante display per visualizzare il file ad esso associato in DocumentViewer, facendo in modo inoltre che ogni oggetto creato abbia come riferimento di origine quel testo. A questo punto per proseguire è necessario premere il tasto Initialize: così facendo una Story Grammar vuota appare in AdvancedTree e si può quindi iniziare l'immissione dei dati. Se l'evento storico su cui si sta lavorando ha come fonte più di un documento si ha la possibilità di selezionare gli altri uno ad uno nel form dedicato e visualizzarli in DocumentViewer per procedere con la loro elaborazione. Una volta completata la procedura basta premere il tasto Save per salvare la Story Grammar così compilata nel database di PC-ACE: da notare che, mentre gli elementi terminali sono salvati subito dopo la loro immissione nelle TextBox, i complex object sono inseriti nel database solo cliccando questo pulsante. Una volta che si è terminato il lavoro sull'evento corrente, se si vuole procedere con la compilazione di una nuova Story Grammar, si può reinizializzare AdvancedTree con una Story Grammar vuota premendo il pulsante Clear.

Un altro vantaggio introdotto da questo layout è la possibilità di inserire il testo selezionato in DocumentViewer direttamente nelle TextBox tramite Drag and Drop, accelerando ancora l'intero processo. In particolare, solo eseguendo l'inserimento con questo metodo sono salvate nel documento di origine le annotazioni corrispondenti agli oggetti inseriti nella grammatica, se la funzione relativa è attivata (per le specifiche di questa funzione vedere la Sezione 5.2.1).

4.3 L'interfacciamento della maschera di inserimento con PC-ACE

DocumentViewer e AdvancedTree sono stati realizzati programmando due controlli ActiveX e costituiscono quindi dei moduli esterni a PC-ACE e al codice VBA sottostante. Le limitazioni di questo linguaggio nel manipolare oggetti, su tutte la mancata implementazione dell'ereditarietà, rendevano impossibile infatti ottenere tutte le funzionalità richieste

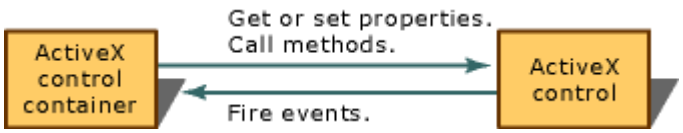


Figura 6 Interazioni tra un controllo ActiveX e il suo contenitore, tratto da [7].

all’interno di Microsoft Access.

Il collegamento con PC-ACE avviene, dunque, attraverso la loro interfaccia pubblica, secondo lo schema generale per i controlli ActiveX mostrato in Figura 6. Anche se un’analisi approfondita dei due controlli, compresa una descrizione dei metodi da loro esposti e gli eventi da loro lanciati, sarà affrontata solo nel Capitolo 5, viene fornita qui una rapida illustrazione di quali routine sono state aggiunge a PC-ACE per realizzare l’interfacciamento.

Il funzionamento di DocumentViewer è relativamente autonomo, non avendo bisogno di alcuna iterazione con il database eccetto il reperimento del percorso del file associato al documento che si vuole visualizzare e l'inserimento degli ID dei simplex object se è attiva la funzione di annotazione.

AdvancedTree invece, essendo il controllo tramite il quale l'inserimento dei dati vero e proprio è realizzato, richiede un’interazione più elaborata con il resto del programma. Escludendo alcuni metodi di comando, sono due i punti di contatto che devono necessariamente essere creati con PC-ACE: l'esportazione della struttura della Story Grammar verso il controllo ActiveX e la gestione dei dati in esso inseriti.

La prima funzionalità è realizzata passando al controllo un array di stringhe, ciascuna rappresentante un elemento della grammatica, formattate nel modo seguente, che riassume tutte le caratteristiche dell'oggetto:

[Level];[ID];[Name];[DataType];[isOptional];[isMultiple];[ExclusiveGroup];

in cui in particolare

- Level indica la profondità dell'oggetto nella Story Grammar rispetto la radice;
- ID è il numero che identifica il tipo di nodo nel database di PC-ACE (il numero è negativo se il nodo è già stato incontrato durante la creazione dell'array e le sue caratteristiche, così come i suoi eventuali figli, sono quindi già state definite);
- ExclusiveGroup indica il gruppo esclusivo di cui fa parte il nodo (è possibile infatti definire per ogni insieme di nodi figli diversi gruppi di nodi mutualmente esclusivi, tra i quali solo uno di essi potrà essere completato);
- Name, DataType, isOptional e isMultiple sono parametri la cui funzione è chiaramente definita dal nome stesso.

La procedura d’inserimento dei dati nel database inizia con la pressione del pulsante Initialize sulla maschera, con cui viene creato un oggetto che rappresenta l’elemento superiore della nuova Story Grammar. Gli oggetti corrispondenti agli altri elementi della grammatica sono aggiunti man mano che si procede con l’inserimento, seguendo le indicazioni fornite tramite gli eventi lanciati da AdvancedTree. Come esposto in precedenza questa collezione di oggetti non è salvata direttamente nel database: solo i simplex object sono inseriti al suo interno nel momento della loro immissione mentre i complex object sono salvati solo al

termine della procedura. La memorizzazione effettiva della Story Grammar, infatti, avviene solo premendo il pulsante Save della maschera, anche se per motivi di sicurezza e di praticità sarebbe preferibile che avvenisse ogni qual volta un nuovo dato viene inserito oppure ad intervalli temporali prestabiliti. Questa differenza di comportamento nel trattare i due tipi di oggetto deriva dalla mancanza di documentazione che affligge la parte centrale del codice di PC-ACE e che ha impedito di individuare la procedura adatta a modificare un oggetto salvato in memoria in tempi ragionevoli.

Anche in questo caso la comunicazione dei dati inseriti da parte del controllo a PC-ACE è basata sul passaggio di un array di stringhe. In questo caso è restituito da AdvancedTree al programma principale tramite un evento lanciato quando s'inserisce un dato in una TextBox, a condizione che sia rispettato il tipo di dato impostato per il simplex object di cui questa costituisce il punto d'inserimento. Le stringhe dell'array sono usate per compiere una visita a partire dalla radice della Story Grammar associata al controllo all'interno di PC-ACE. In questo caso hanno la struttura

[AlreadyDone];[isSimplex];[ParentID];[ID];[NodeOrder];[DataType];[Value];

dove, in particolare, AlreadyDone indica se l'oggetto che si sta trattando è già stato creato in memoria e NodeOrder la sua posizione all'interno della lista dei figli dello stesso nodo con lo stesso ID, mentre gli altri parametri sono di facile comprensione. L'ID dell'oggetto padre, necessario per creare in memoria gli oggetti associati agli elementi della grammatica, e il tipo di dato potrebbero essere reperiti direttamente dal codice VBA ma sono forniti lo stesso come misura di sicurezza, per evitare che un passaggio di parametri errati causi un errore possibilmente fatale. Solo l'ultima stringa dell'array provoca un effettivo inserimento di dati all'interno del database di PC-ACE: al raggiungimento del simplex object prima viene effettuata una ricerca nel dizionario per verificare se il valore indicato da Value è presente, per essere aggiunto solo in caso negativo, e poi viene creato nel database il simplex corrispondente.

L'altro importante punto di collegamento con il codice VBA è costituito dalla routine di eliminazione degli oggetti: è infatti possibile in ogni momento rimuovere un oggetto in AdvancedTree selezionando l'opzione da un menù contestuale associato ad ogni oggetto. Con questa procedura sono restituiti due array. Uno di essi serve per segnalare a DocumentViewer l'ID di tutti i simplex object che sono stati eliminati dalla Story Grammar corrente affinché proceda con l'eliminazione delle annotazioni a loro associate (ovviamente se la funzione relativa in DocumentViewer è attiva). L'altro è un array di comando il cui compito è procedere all'eliminazione effettiva degli oggetti nella Story Grammar associata al controllo all'interno di PC-ACE. Le stringhe di comando hanno la seguente struttura:

[IsSimplex];[ParentID];[ID];[NodeOrder];

dove tutti i parametri sono di facile comprensione. In analogia con la procedura di inserimento dei dati l'array serve per compiere una visita all'interno dell'albero a partire dalla radice e solo l'ultima stringa dell'array indica quale nodo eliminare, sia esso un simplex od un complex.

5 I controlli ActiveX

In questo Capitolo verranno descritti con maggiore dettaglio i due controlli ActiveX usati nella maschera di inserimento sviluppata nel corso di questa tesi. Nella prima sezione sono introdotti la tecnologia ActiveX e l'ambiente di sviluppo. Successivamente si procederà con l'illustrazione del funzionamento di DocumentViewer e AdvancedTree, tramite la descrizione delle loro interfacce pubbliche e di alcuni aspetti salienti nella loro programmazione. Nelle ultime due sezioni si discuteranno i problemi noti e le possibili espansioni.

5.1 La tecnologia ActiveX e il .NET Framework

La tecnologia ActiveX fu introdotta dalla Microsoft nel 1996 per facilitare l'estensione delle funzionalità delle applicazioni in ambiente Windows. Rappresenta tuttora l'unico modo per aggiungere dei controlli grafici personalizzati nelle applicazioni nel pacchetto Office come Microsoft Access e per questo motivo è stato necessario programmare AdvancedTree e DocumentViewer come controlli ActiveX.

Questa tecnologia deriva da due standard precedenti con cui è spesso confusa, ossia OLE (Object Linking and Embedding), il protocollo che per primo ha permesso la realizzazione di documenti compositi in ambiente Windows (consentendo ad esempio l'apertura di una tabella di Microsoft Excel in Microsoft Word), e COM (Component Object Model), una tecnologia basata su OLE 2.0 che permette la comunicazione tra processi e la creazione dinamica di oggetti da parte di qualsiasi linguaggio da cui è supportata. Questo collegamento è realizzato derivando l'interfaccia IUnknown, che definisce l'accesso ad una tabella di puntatori alle funzioni che costituiscono l'interfaccia pubblica vera e propria del controllo, mentre per realizzare componenti più complessi può essere richiesta l'implementazione di altre interfacce: ad esempio l'interfaccia IOleObject deve essere realizzata da quegli oggetti creati per essere incapsulati in un contenitore, come i controlli ActiveX [8]. I due protocolli OLE e COM sono importanti perché da essi derivano alcuni requisiti: ogni ActiveX è un oggetto COM e richiede dunque una GUID (Globally Unique Identifier) per assicurare la sua unicità nel sistema e l'inserimento di alcune voci nel registro di sistema per specificare quali interfacce definiscono i suoi metodi pubblici e dove è collocata, rendendo necessaria una procedura d'installazione.

Dal 2002 la piattaforma COM è stata superata a favore del .NET Framework, nato in seno alla Microsoft come alternativa proprietaria al linguaggio Java. Il suo funzionamento è

simile a quello di questo linguaggio. Anche nella piattaforma .NET il codice, che può essere scritto in vari linguaggi di programmazione come il C++ o il C#, viene compilato in un assembly scritto in linguaggio CIL (Common Intermediate Language) che viene poi eseguito da una macchina virtuale. In questo modo i file compilati sono indipendenti sia dalla piattaforma hardware sia da quella software, anche se l'esecuzione in ambienti diversi da Windows è possibile solo sfruttando pacchetti software sviluppati da società terze.

Tra la tecnologia COM e l'ambiente .NET è stata mantenuta, però, una sorta di retrocompatibilità: in questo modo è stato possibile realizzare i due controlli DocumentViewer e AdvancedTree come librerie .NET esponendole come controlli ActiveX tramite la funzionalità di Interoperabilità COM. Per rendere i due controlli compatibili con le applicazioni del pacchetto Office come Microsoft Access sono necessari alcuni accorgimenti addizionali oltre ad abilitare l'interoperabilità all'interno del programma di sviluppo. È necessario, infatti, che la classe principale del controllo:

1) estenda la classe System.Windows.Forms.UserControl;
2) implementi due interfacce, una indicante i metodi e le proprietà esposte dall'ActiveX e l'altra contenente i riferimenti agli eventi che il controllo può lanciare;
3) includa dei metodi che provvedano alla registrazione nel Registro di Sistema di Windows e alla cancellazione del controllo da esso durante l'installazione.

La retrocompatibilità non è, però, completa: alcune delle limitazioni esposte nella Sezione 5.4 dipendono con tutta probabilità proprio da questo.

Le classi che costituiscono il .NET Framework sono più ricche, più documentate e più facilmente estendibili rispetto a quelle costituenti l'API (Application Programming Interface) di Windows, ossia la libreria che si può sfruttare per la costruzione delle ActiveX, soprattutto per la parte grafica. Si è scelto di implementare i due controlli nell'ambiente .NET in quanto sarebbe stato molto più complesso estendere i controlli della Windows API aggiungendo le funzionalità richieste rispetto al lavoro richiesto operando su Windows Forms, la libreria grafica del .NET Framework. Infine l'implementazione di tutte le interfacce richieste da un controllo da usare in Microsoft Access sarebbe stata difficile da realizzare senza poter sfruttare delle procedure guidate o dei template già esistenti, disponibili però solo nelle versioni a pagamento di Microsoft Visual Studio o di altri IDE (Integrated Development Environment) equivalenti.

AdvancedTree e DocumentViewer sono quindi stati scritti in C#, un linguaggio di programmazione simile ai più noti Java e C++ sviluppato dalla Microsoft all'interno del .NET Framework, progettandoli per .NET Framework 4.0 tramite Microsoft Visual C# 2010 Express Edition, la versione gratuita dell'IDE ufficiale per questo linguaggio, e testandoli nella prima fase di sviluppo usando una utility dedicata presente nella suite Microsoft Visual Studio, di cui Visual C# fa parte.

5.2 DocumentViewer

DocumentViewer assolve due diverse funzioni all'interno di PC-ACE: può essere impiegato infatti sia per visualizzare i documenti nei vari formati supportati dal programma sia come annotatore rudimentale, consentendo il salvataggio nel corpo del testo di origine di tag indicanti il tipo di simplex e l'ID nel database delle parole inserite nel database. Strutturalmente è molto semplice, essendo composto da tre classi e una interfaccia contenute tutte in un unico namespace.

La visualizzazione dei documenti è realizzata sfruttando vari controlli grafici presenti all'interno della piattaforma .NET, per i quali DocumentViewer si comporta come semplice contenitore provvedendo alla loro creazione, alla loro corretta inizializzazione e alla loro eliminazione. Sono supportati i file di testo con le estensioni più comuni (txt, rtf e doc), diverse codifiche di immagini (bmp, jpg, gif, tiff e png), le pagine in formato HTML e i documenti in formato pdf. Da notare che la visualizzazione dei file di Microsoft Word, ossia quelli con estensione doc, è possibile solo sfruttando alcune librerie fornite con questo applicativo, richiedendo quindi la sua presenza sul computer. La visualizzazione dei file in formato pdf, invece, è possibile solo usando un visualizzatore esterno al Framework .NET, non essendo questa tecnologia supportata in modo nativo dalla Microsoft. Per contenere il

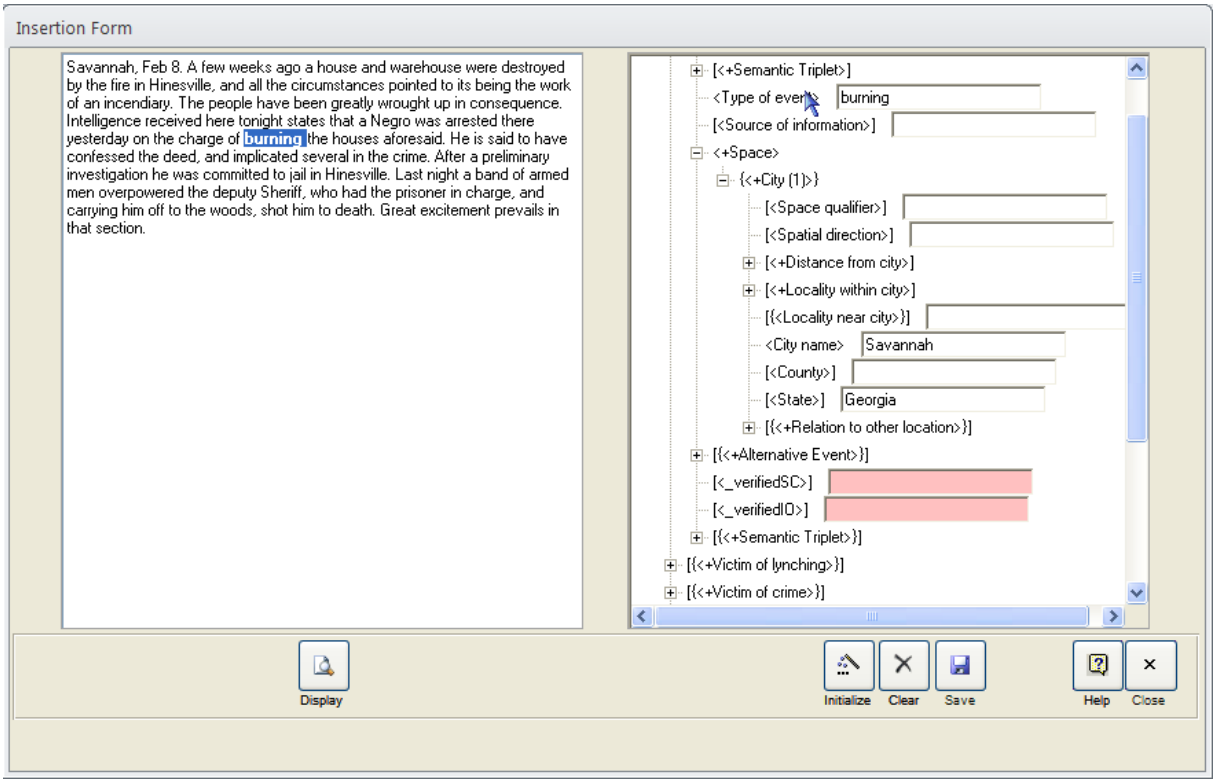


Figura 7 Sulla sinistra DocumentViewer in modalità Annotatore.

numero di dipendenze esterne si è scelto di usare a questo scopo l'Adobe Acrobat ActiveX Control, installato di norma insieme all'Adobe Acrobat Reader, il più diffuso lettore di file in formato pdf che si assume essere presente in ogni macchina in cui è eseguito PC-ACE.

5.2.1 L'annotatore

La funzione di annotazione presente in DocumentViewer consente di salvare nel testo del documento su cui si sta operando informazioni riguardanti la collocazione nel database delle parole estratte per l'inserimento, permettendo sia un controllo visivo durante l'immissione su quali dati sono stati estratti sia una analisi in un secondo momento della procedura di immissione. Nella versione attuale soffre di due limitazioni importanti; la prima dipende dall'implementazione e limita il suo utilizzo esclusivamente a file di testo non formattati salvati con l'estensione paf (PC-ACE Annotator File). La seconda dipende invece dai limiti nella procedura d'inserimento descritti nel Capitolo 4, a causa delle quali è possibile il salvataggio solo delle informazioni relative ai simplex object perché quelle relative ai complex object a loro superiori nella struttura della Story Grammar, nel caso in cui essi fossero riducibili ad una parte specifica del testo, non sono note al momento della creazione.

Il funzionamento dell'annotatore è molto semplice. Per aggiungere una nuova annotazione è sufficiente invocare il metodo specifico per la sua creazione, che controlla anche che ci sia un'effettiva corrispondenza tra il testo selezionato e quello che si vorrebbe annotare, dopo aver selezionato una porzione del documento. Questo metodo è chiamato in PC-ACE durante la procedura d'inserimento dei dati. Perché sia eseguito in modo corretto è necessario completare le TextBox di AdvancedTree tramite Drag and Drop: solo così si ha la certezza di una corrispondenza tra dato selezionato in DocumentViewer e dato inserito in AdvancedTree. Inoltre, AdvancedTree lancia un evento al passaggio del cursore del mouse sopra i nodi così compilati tramite il quale le annotazioni a loro associate sono evidenziate nella schermata dell'annotatore, mostrando esplicitamente il collegamento tra gli elementi inseriti nella grammatica e la loro origine nel documento (vedere la Figura 7).

Le annotazioni, memorizzate in una lista di oggetti durante la visualizzazione del file, sono salvate nel corpo del file paf solo al momento della chiusura del documento. Sono inserite tramite l'aggiunta al testo di un *tag* con due attributi, id e db, con la seguente sintassi:

<simplex id=[number] db =[number]>[text]</simplex>

dove il numero associato ad id indica il tipo di simplex, il numero associato a db indica l'ID con cui l'oggetto è stato inserito nel database mentre [text] è il testo che si è inserito in PC-ACE.

Alla riapertura del documento in DocumentViewer i tag sono estratti dal testo e salvati in memoria e sono gestiti alla pari di quelli creati in quella sessione di lavoro. L'estrazione delle annotazioni, compiuta sfruttando un'espressione regolare, e la procedura adottata per il

loro salvataggio impongono un altro limite al controllo: i tag non possono essere annidati ma devono essere creati separati l'uno dall'altro affinché queste procedure vadano a buon fine.

5.2.2 L'interfaccia pubblica

```
public interface IViewerControl
{
    void AnnotatorCreateAnnotation(string data, int simplexID, int
databaseID);

    void AnnotatorHighlight(int databaseID);

    bool AnnotatorIsActive();

    void AnnotatorRemoveAnnotation(int databaseID);

    void AnnotatorRemoveHighlight(int databaseID);

    void CloseDocument();

    void DocumentSource(string path);

    bool HasDocument();

    bool SupportFileType(string fileExtension);
}
```

Figura 8 Interfaccia pubblica di DocumentViewer**.**

Nella Figura 8 è riportata l'interfaccia pubblica di DocumentViewer; i metodi che la compongono sono divisi in due gruppi, quelli di utilizzo generale e quelli specifici alla gestione dell'annotatore. In particolare:

- `AnnotatorCreateAnnotation` crea una nuova annotazione nel documento corrente; `simplexID` e `databaseID` sono i dati salvati nell'annotazione, indicando rispettivamente il tipo di simplex e il suo riferimento nel database, mentre il parametro `data` è usato per fare un confronto tra il testo selezionato e quello che è stato inserito nel database.
- `AnnotatorHighlight` evidenzia a video le annotazioni contrassegnate da `databaseID`.
- `AnnotatorIsActive` restituisce `true` se l'annotatore è attivo.
- `AnnotatorRemoveAnnotation` elimina la prima annotazione contrassegnata da `databaseID`.
- `AnnotatorRemoveHighlight` rimuove l'evidenziazione dalle annotazioni contrassegnate da `databaseID`.

- `CloseDocument` chiude il documento corrente. Per un corretto funzionamento del controllo va invocato prima di cambiare documento e prima della chiusura del form in cui è posto. Provvede inoltre al salvataggio delle annotazioni.
- `DocumentSource` visualizza il file localizzato nel percorso indicato da `path`. Lancia un'eccezione se c'è già un documento aperto o se il file non è di tipo valido.
- `HasDocument` ritorna `true` se è visualizzato un documento.
- `SupportFileType` ritorna `true` se `fileExtension` è supportata da DocumentViewer.

5.3 AdvancedTree

AdvancedTree è il controllo ActiveX che consente a PC-ACE di avere un'interfaccia grafica più interattiva nella procedura d'inserimento dei dati. L'estrema flessibilità consentita nella definizione delle Story Grammar e la necessità di avere a disposizione un alto numero di funzioni ha portato alla creazione di un controllo più complesso del precedente. Il nuovo controllo è composto da due interfacce e nove classi divise in due namespace (vedere la Figura 9): mentre in AdvancedTree sono contenute le classi legate al funzionamento vero e proprio del controllo, nel namespace PC_ACE sono contenute le due classi legate alla gestione dell'esemplare di Story Grammar fornito all'inizializzazione del controllo.

Esso visualizza in modo semplice e intuitivo la Story Grammar che si sta completando: si basa infatti su un controllo ad albero che rappresenta i suoi oggetti tramite una lista indentata. Ogni elemento o nodo di cui è costituito può avere un qualsiasi numero di figli, che si possono visualizzare espandendo il nodo padre o nascondere collassandolo. Come sarà illustrato in seguito la classe base presente all'interno del .NET Framework è stata estesa aggiungendo varie funzionalità. Guardando le figure precedenti si nota immediatamente la modifica principale, ossia l'aggiunta delle TextBox agli elementi terminali della grammatica che rendono possibile l'immissione dei dati.

5.3.1 Struttura e funzionamento

Tutte le classi appartenenti al namespace AdvancedTree estendono classi già presenti nel .NET Framework all'interno del namespace System.Windows.Forms, che contiene tutte le classi della GUI (Graphical User Interface) nativa dell'ambiente .NET. Tutte le nuove classi aggiungono delle funzionalità agli elementi grafici usati nella realizzazione del controllo. Nello specifico:

- AdComboBox estende la classe ComboBox, la classe che realizza la casella combinata che permette la scelta di un'opzione da un menù a tendina.

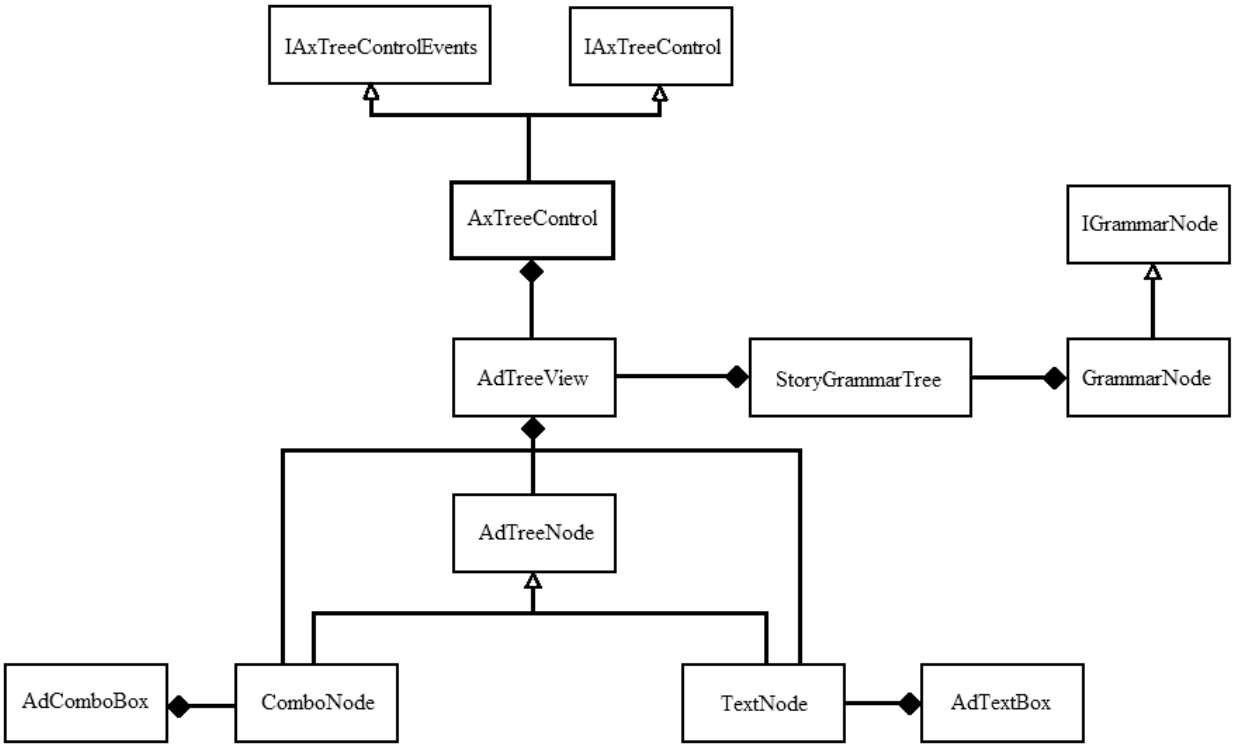


Figura 9 Diagramma delle relazioni tra le classi e le interfacce che compongono AdvancedTree

- AdTextBox estende TextBox, la classe che realizza la casella di testo tramite cui è possibile inserire dei dati nel controllo.
- AdTreeNode estende la classe TreeNode, la classe che rappresenta un nodo visualizzato tramite il controllo TreeView. In esso sono salvate tutte le caratteristiche dell'elemento della grammatica che rappresenta tramite l'implementazione dell'interfaccia IGrammarNode che definisce un oggetto della Story Grammar così come sono stati introdotti nella Sezione 2.2.1.
- AdTreeView estende la classe TreeView, la classe che consente la visualizzazione di dati organizzati in modo gerarchico tramite una rappresentazione con una lista indentata. Ogni elemento è un'istanza della classe AdTreeNode o delle classi da lei derivate e il loro insieme rappresenta la Story Grammar che si sta inserendo nel database.
- AxTreeControl estende la classe UserControl, costituendo quindi la classe principale del controllo e implementando le due interfacce descritte in dettaglio nella Sezione 5.3.2.
- ComboNode estende la classe AdTreeNode associando al nodo una AdComboBox; viene usata per rappresentare tutti i nodi in cui si deve compiere una scelta tra elementi mutualmente esclusivi.
- TextNode estende la classe AdTreeNode associando al nodo una AdTextBox; viene usata per rappresentare tutti i nodi terminali della grammatica.

All'interno del namespace PC_ACE sono presenti invece solo due classi:

- GrammarNode, le cui istanze rappresentano la definizione di un elemento della grammatica tramite l'implementazione dell'interfaccia IGrammarNode.
- StoryGrammarTree, le cui istanze rappresentano la definizione della struttura di una intera Story Grammar.

Le classi principali del controllo sono tre:

- StoryGrammarTree: mette a disposizione del controllo una copia vuota di una Story Grammar usata come master nelle operazioni di creazione dei nodi.
- AdTreeView: contiene tutte le funzioni legate all'interazione dell''utente con l'albero.
- AxTreeControl: cura l'interfacciamento del controllo con l'esterno, in questo caso specifico con PC-ACE.

Il collegamento con il database descritto nella sezione 4.3 coinvolge quindi solo le prime due classi con la terza che agisce semplicemente da tramite. Attraverso un metodo di StoryGrammarTree si provvede, infatti, a convertire l'array di stringhe che rappresenta una Story Grammar in una struttura ad albero ed a inizializzarla per l'uso nel controllo vero e proprio. Gli array di comando, che avviano l'inserimento o l'eliminazione di un elemento della grammatica che si sta compilando nel database, sono generati invece a partire da due eventi lanciati da AxTreeControl.

Per comprendere correttamente il funzionamento del controllo è necessario fornire una descrizione più approfondita di AdTreeView rispetto alle altre componenti del controllo. Questa classe infatti, oltre a presentare alcuni metodi specifici per l'inizializzazione dell'albero e per la gestione dei suoi elementi, sovrascrive numerosi metodi di TreeView responsabili della gestione degli eventi: in questo modo sono state aggiunte le varie funzionalità che distinguono AdTreeView dalla sua classe base. In particolare, ora è possibile eseguire le seguenti operazioni:

- espandendo un elemento sono create e visualizzate le AdTextBox per ogni suo figlio ora visibile sullo schermo.
- collassando un elemento le AdTextBox visualizzate nei suoi figli sono nascoste.
- cliccando con il tasto sinistro del mouse su un elemento possono verificarsi due situazioni distinte: se il nodo è un ComboNode allora viene visualizzato il menù a tendina ad esso associato mentre vengono creati i suoi figli se la loro inizializzazione non era stata completata perché questo poteva portare ad un fenomeno ricorsivo, possibile nel caso in cui un oggetto complesso sia figlio di sé stesso. Ad esempio una tripletta semantica può avere come nodo figlio una "tripletta alternativa", che costituisce un alias del suo oggetto padre (gli alias sono stati descritti nella Sezione 3.1). Se non s'interrompesse l'inizializzazione questa continuerebbe all'infinito, in

quanto ogni volta viene creato il nodo “tripletta semantica” figlio si dovrebbe procedere all’inizializzazione del nodo “tripletta semantica” figlio di questo.

- cliccando con il tasto destro del mouse su un elemento viene visualizzato un menù contestuale che consente la scelta tra varie operazioni da compiere sul nodo; nella versione attuale si può scegliere se eliminarlo o se procedere con la sua moltiplicazione nel caso consenta copie multiple e i suoi figli obbligatori siano stati completati.
- spostando il mouse sopra ad un elemento terminale viene lanciato un evento che consente all'annotatore di evidenziare il testo ad esso associato.

In questa stessa classe sono contenuti anche i metodi che gestiscono la selezione di un’opzione in un ComboNode e l'inserimento dei dati in un TextNode. Nel primo caso il nodo è sostituito con il sottoalbero corrispondente alla scelta mentre nel secondo l’inserimento viene notificato al database tramite un evento dopo aver eseguito un controllo per verificare che il tipo di dato corrisponda con l'oggetto specificato. Questo controllo è comunque possibile per l’utente anche durante l'immissione delle informazioni, in quanto le AdTextBox hanno un colore di sfondo diverso a seconda del tipo di dato associato al nodo a cui sono assegnate.

5.3.2 L'interfaccia pubblica

In Figura 10 è riportata l'interfaccia pubblica di AdvancedTree, costituita da due interfacce che definiscono una i metodi e l'altra gli eventi lanciati dal controllo. In particolare:

- `AddEmptyGrammarTree` aggiunge una Story Grammar vuota al controllo.
- `DeleteTree` elimina dal controllo la Story Grammar visualizzata in quel momento.
- `InitializeGrammar` inizializza la grammatica tramite l'array di stringhe `grammar`, la cui sintassi è definita nella Sezione 4.3. Il parametro è dichiarato come object a causa di un problema di compatibilità con VBA.
- `SetDateFormat` sovrascrive il formato della data predefinito (DD/MM/YYYY) con quello contenuto in `format`, ritornando `true` se la procedura va a buon fine. Questo viene usato per controllare che le date inserite siano nel formato stabilito in quel momento da PC-ACE.
- `SetSimplexDatabaseID` associa all'ultimo elemento inserito nel database tramite l'evento `OnDataEntered` la sua `XrefID` in memoria.
- `SetTimeFormat` sovrascrive il formato del tempo predefinito (HH.MM.SS) con quello contenuto in `format`, ritornando `true` se la procedura va a buon fine. Questo viene usato per controllare che i tempi inseriti siano nel formato stabilito in quel momento da PC-ACE.

```
public interface IAxTreeControl
{
    void AddEmptyGrammarTree();

    void DeleteTree();

    void InitializeGrammar(object grammar);

    bool SetDateFormat(String format);

    void SetSimplexDatabaseID(int XrefID);

    bool SetTimeFormat(String format);
}
public interface IAxTreeControlEvents
{
    void OnDataDeleted(ref String[] deleteData, ref int[]
deletedSimplexID);

    void OnDataEntered(ref String[] newData);

    void OnMouseEnterNode(int DatabaseID);

    void OnMouseLeaveNode(int DatabaseID);
}
```

Figura 10 Le due interfacce pubbliche di AdvancedTree: IAxTreeControl definisce i metodi pubblici mentre IAxTreeControlEvents definisce gli eventi che possono essere lanciati dal controllo.

- `OnDataDeleted` è l'evento che notifica a PC-ACE che un oggetto è stato eliminato; `deleteData` è l'array di comando usato nella procedura di eliminazione descritta nella Sezione 4.3 mentre `deletedSimplexID` contiene gli ID di tutti i simplex eliminati in questa procedura, consentendo a DocumentViewer di eliminare le annotazioni a loro associate se questa funzione è attiva.
- `OnDataEntered` è l'evento che notifica a PC-ACE che un dato è stato inserito; `newData` è l'array di comando usato nella procedura di inserimento descritta nella sezione 4.3.
- `OnMouseEnterNode` è l'evento che notifica che il puntatore del mouse è entrato in un nodo del controllo; serve per evidenziare l'annotazione relativa al nodo passando `databaseID` a DocumentViewer.
- `OnMouseLeaveNode` è l'evento che notifica che il puntatore del mouse è uscito da un nodo del controllo; serve per eliminare l'evidenziazione relativa al nodo passando `databaseID` a DocumentViewer.

Da notare come gli array di comando nei parametri degli eventi devono essere passati esplicitamente per riferimento affinché possano essere utilizzati dal codice VBA.

5.4 Problemi noti

I problemi e i limiti che a due controlli hanno manifestato in fase di testing si possono dividere in tre gruppi: quelli propri dell'architettura e delle estensioni utilizzate, quelli legati alla grafica e quelli dovuti alla mancanza di documentazione del codice su cui si basa PC-ACE, a cui si è già in parte accennato nella Sezione 4.3.

Nel primo gruppo si collocano dei problemi diversi tra loro che presentano però un denominatore comune: dipendono dalla strategia implementativa e possono dunque essere risolti usando degli strumenti alternativi. Uno di questi si nota solo operando con i due controlli nella modalità di design della maschera: si vede subito come essi appaiono diversi rispetto ai controlli ActiveX standard in quanto le loro dimensioni a video sono fissate e non espongono esplicitamente metodi, eventi e proprietà come gli altri controlli. La spiegazione più plausibile a questo comportamento deriva dalla non completa retrocompatibilità tra gli oggetti COM e le librerie .NET accennata nella Sezione 5.1. Oltre a questi limiti, che sono comuni ad entrambi i controlli, DocumentViewer soffre di un problema nel visualizzare i documenti nel formato di Microsoft Word: infatti le immagini in essi contenute non sono visualizzate perché la procedura che consente la loro elaborazione estrae solo il testo da questi documenti.

S’incontra poi un problema relativo alla resa grafica di AdvancedTree legato allo scorrimento verticale o orizzontale del controllo ad albero. Le AdTextBox utilizzate in AdvancedTree non fanno infatti parte dell'area grafica propria dei nodi di AdTreeView, ma sono solo associate ad essi: come risultato la classe madre non aggiorna la loro posizione a video quando si spostano le barre di scorrimento o si utilizza la rotellina del mouse per scorrere verticalmente il controllo. Il metodo adottato per ovviare a questo problema, ossia l'intercettazione dei messaggi relativi agli scorrimenti inviati dall'ambiente .NET, non consente un aggiornamento fluido delle posizioni per via della natura discreta e non continua che li caratterizza. Questo provoca spesso uno scintillio delle AdTextBox durante il movimento o un loro posizionamento leggermente sbagliato alla fine del movimento (che può essere risolto facilmente da parte dell'utente espandendo e collassando un nodo qualsiasi).

L'ultima classe di problemi dipende dal codice VBA preesistente in PC-ACE con cui i due controlli si devono interfacciare. Alcune funzioni avanzate, come il salvataggio dei complex object man mano che sono creati, il Copia e Incolla di oggetti da Story Grammar già esistenti a quella che si sta completando in AdvancedTree o l'apertura di Story Grammar già presenti nel database, non sono di immediata realizzazione a causa di difficoltà oggettive nella comprensione del funzionamento delle routine coinvolte. Una loro riprogrammazione da zero era stata presa in considerazione ma sarebbe stata ugualmente complessa e avrebbe

comportato la duplicazione indesiderata di queste procedure, complicando ulteriormente la manutenzione del codice.

5.5 Ampliamenti futuri

Esistono diverse funzionalità che in futuro potrebbero essere inserite per migliorare l'utilizzabilità dei due controlli. Due sono gli aspetti su cui si deve sicuramente intervenire: come prima cosa si dovrebbe procedere con un'estensione di AdvancedTree con lo scopo di risolvere i problemi e superare i limiti relativi all'interfacciamento con PC-ACE incontrati nel suo sviluppo illustrati nella sezione precedente. Il secondo obiettivo dovrebbe essere l'espansione della funzione di annotazione rendendola disponibile per tutti i tipi di documenti e non solo per quelli salvati in un formato dedicato come avviene in questo momento. Per poterlo realizzare bisognerebbe probabilmente abbandonare l'approccio adottato in questa sede per il salvataggio dei tag e procedere con la loro memorizzazione in un file separato o direttamente nel database di PC-ACE. La memorizzazione delle annotazioni relative ai file di immagine e ai file pdf richiederebbe quasi sicuramente un cambiamento del metodo con cui sono visualizzate, comportando nel caso dei file pdf un cambiamento anche del visualizzatore utilizzato.

Più complesso sarebbe estendere l'utilizzo di AdvancedTree e di DocumentViewer ad altri ambiti all'interno di PC-ACE o ad altre tipologie di analisi dei testi che potrebbe essere d'interesse nell'ambito della ricerca sociologica, ad esempio l'analisi retorica dei documenti. Questa limitazione riguarda in particolar modo AdvancedTree: DocumentViewer è stato realizzato in modo più flessibile e con l'obiettivo di essere usato per visualizzare i documenti in tutti i contesti previsti dal programma. Anche l'annotatore può essere facilmente esteso per consentirgli di importare dall'esterno la sintassi usata nei suoi tag, permettendogli quindi di adattarsi alle più diverse operazioni di annotazione. Il codice di AdvancedTree invece non è direttamente portabile per altri utilizzi poiché le sue classi, in particolare AdTreeView, sono legate in modo molto stretto alla necessità di visualizzare una Story Grammar. Le varie soluzioni tecniche adottate nella creazione di un controllo d'inserimento ad albero, ad esempio l'inclusione all'interno dei suoi nodi di TextBox e ComboBox e il collegamento dello stesso con un oggetto che ne fa da esemplare per la sua realizzazione, si potrebbero comunque esportare e riutilizzare con facilità. In questo caso AdvancedTree diventerebbe simile a DocumentViewer, ossia un contenitore di varie interfacce d'inserimento diverse attivate di volta in volta a seconda della necessità.

6 Automatizzazione dell'inserimento dei dati

Il miglioramento delle prestazioni ottenuto con la nuova interfaccia rappresenta un primo passo verso una fase d'inserimento dei dati più "leggera" dal punto di vista dell'utilizzatore finale. Si potrebbe ottenere un consistente aumento della velocità dell'acquisizione dei dati in PC-ACE se fosse possibile eseguire quest'operazione in modo automatizzato, limitando l'intervento umano solo all'inizializzazione della procedura stessa e al controllo delle informazioni così estratte. Tra le varie tecniche presenti nel campo del *Natural Language Processing*, ovvero l'area dell'informatica che si occupa dell'elaborazione elettronica di testi scritti in linguaggio umano o naturale, l'*Information Extraction* sembra essere quella che più adatta a risolvere questo problema. In questo capitolo saranno quindi introdotte le basi teoriche di questo metodo di elaborazione dei documenti per poi procedere ad una discussione sulla sua applicabilità alla Quantitative Narrative Analysis e a PC-ACE.

6.1 Introduzione all'Information Extraction

L'Information Extraction [9] è una tecnica basata sull'analisi del linguaggio naturale che mira ad estrarre parti specifiche, siano esse parole o locuzioni, da un testo fornito come input combinandole in uno schema fissato all'inizio dell'operazione e caratteristico dell'evento trattato. È molto diversa, dunque, dall'*Information Retrieval*, la tecnologia usata per scopi d'indicizzazione in applicazioni come i motori di ricerca, che non restituiscono le informazioni specifiche cui l'utente è interessato ma i testi più rilevanti nell'ambito della ricerca effettuata. Si presenta quindi come una tecnica che produce un risultato molto simile a quanto ottenuto manualmente durante la compilazione di una Story Grammar in PC-ACE.

Nel senso attuale del termine l'Information Extraction nasce dal lavoro svolto durante le Message Understanding Conferences (MUC), sponsorizzate dal governo statunitense dalla fine degli anni '80 al 1997. Al termine di queste conferenze si è arrivati ad una sua definizione canonica; essa viene infatti divisa in cinque attività fondamentali distinte che la identificano completamente [10]:

1) La *Named Entity Recognition*, in cui sono individuate e classificate nel testo entità come nomi propri, luoghi, date etc. Tra le varie attività è quella con il più alto grado di affidabilità, raggiungendo una precisione vicina al 95%.
2) La *Coreference Resolution*, in cui sono individuate le relazioni tra le entità rilevate nella prima fase. Si divide nella risoluzione delle anafore e dei nomi propri, che stabiliscono rispettivamente a cosa si riferiscono i pronomi presenti nel testo e se la

stessa entità è indicata con parole diverse. Questa fase è importante perché permette di associare tutte le informazioni riferite ad una stessa entità anche se sono sparse in vari punti del testo in esame. Al contrario del precedente è un processo impreciso, soprattutto per quanto riguarda la risoluzione delle anafore, e presenta un'affidabilità media del 50-60%.

3) La *Template Element Construction*, in cui sono aggiunte informazioni descrittive ai risultati della prima fase usando le informazioni fornite nella seconda. In sistemi buoni questa fase presenta un'affidabilità attorno all'80%.
4) La *Template Relation Construction*, in cui sono individuate le relazioni tra le entità costruite nella terza fase. In questa attività si raggiunge di norma una precisione vicina al 75%.
5) La *Scenario Template Production*, in cui le informazioni ottenute dalla terza e la quarta fase sono compresse all'interno di schemi per gli eventi determinati in precedenza. Questa è la parte più complessa dell'intero processo ed ha un'affidabilità media del 60%.

Questa suddivisione non va intesa rigidamente: è infatti legata in parte alle attività e agli obiettivi assegnati ai team in gara nelle varie conferenze. Una classificazione alternativa è costituita, ad esempio, da quella adottata dal progetto Automatic Content Extraction (ACE), che si può considerare il successore del MUC anche se al contrario di questa non rende disponibili all'esterno documentazione e risultati. In ACE le prime due attività sono combinate in una singola *Entity Detection and Tracking*, la seconda coppia di attività nella *Relation Detection and Tracking* e l'ultima fase è rinominata *Event Detection and Characterisation*. Si può facilmente notare, al di là della denominazione, come la classificazione sia equivalente. In seguito sarà utilizzata la prima perché mette più in risalto le diverse operazioni che devono essere compiute per completare l'estrazione dei dati.

Per illustrare meglio l'applicazione di questa tecnica e la correlazione tra le varie fasi in cui viene scomposta si consideri la seguente frase:

"Ieri l'anziano proprietario della Tabaccheria Le Rose è stato derubato da rapinatori sconosciuti che si sono dileguati a bordo di un'auto rossa."

Applicando il metodo descritto in precedenza a questo periodo nella Named Entity Recognition si individuano come entità del discorso "*ieri*", "*proprietario*", "*Tabaccheria Le Rose*", "*rapinatori*" e "*auto*", nella Coreference Resolution si scopre che "*che*" si riferisce ai "*rapinatori*", nella Template Element Construction si assegnano "*anziano*" a "*proprietario*", "*sconosciuti*" a "*rapinatori*" e "*rossa*" ad "*auto*", nella Template Relation Construction si individua che la "*Tabaccheria Le Rose*" appartiene all'"*anziano proprietario*" e infine nella Scenario Template Production si ricava che c'è stata una rapina con successiva fuga in cui le varie entità sono coinvolte, e si compilano gli schemi associati a questi due eventi.

Bisogna comunque fare due precisazioni su questa tecnica e sui numeri portati come esempio. Innanzitutto le varie attività sopra descritte sono legate al dominio d'interesse dell'utente finale, anche se ognuna in modo diverso: se la Named Entity Recognition è la più indipendente, la Scenario Template Production è basata completamente sullo scenario d'interesse. Riferendosi poi sempre all'esempio portato in precedenza, per iniziare l'analisi di quella frase il ricercatore avrebbe dovuto impostare il sistema per analizzare eventi riguardanti "rapine, omicidi, crimine organizzato, fughe, …", stabilendo che le entità d'interesse in questi domini sono "persone, aziende, banche, mezzi di trasporto, località, organizzazioni, tempi, …" e fornendo dei collegamenti tra queste entità sia di tipo descrittivo sia di tipo relazionale. Il completamento dei primi due deve essere svolto manualmente, creando i template da completare per ogni tipologia di evento e fornendo il lessico relativo alle entità di interesse. Il terzo passo, invece, può essere svolto anche in modo automatico tramite l'apprendimento da un *corpus* annotato, ossia un insieme di testi sui quali siano già state compiute operazioni di annotazioni per indicare la funzione grammaticale o logica di ogni parola.

I numeri forniti per l'affidabilità delle varie fasi (riportati in [10]) non devono comunque trarre in inganno. Anche operatori umani addestrati al compito non riescono a raggiungere il 100% di affidabilità nell'estrazione delle informazioni a causa di sviste o interpretazioni errate; in particolare nell'ultima fase, che è senza dubbio la più importante per la buona riuscita dell'operazione, un umano può arrivare nel caso pessimo ad un'accuratezza attorno all'80%.

6.2 Un'architettura per un programma di Information Extraction

I primi tentativi di realizzare un software per compiere Information Extraction erano basati sulla riduzione di questa tecnica ad un mero problema di comprensione dei testi. L'analisi veniva dunque compiuta applicando ad ogni frase dei documenti dati come input un algoritmo di *parsing* generico che le suddivideva nei loro elementi logici da cui venivano poi estratte le informazioni tramite degli algoritmi specifici al dominio di interesse [11]. Questo metodo presenta però vari problemi che portavano a risultati non soddisfacenti. In primo luogo anche le grammatiche più ricche non coprivano completamente tutte le regole incontrate in testi comuni come gli articoli di giornale (come esempio si può considerare che sono state ricavate 17540 regole grammaticali diverse da un corpus di articoli del Wall Street Journal [12]), costringendo gli sviluppatori a programmare *parser* rigidi, portatori quindi di errori. Le frasi nei testi considerati, inoltre, sono spesso lunghe oltre le cinquanta parole e in diversi casi anche oltre le cento. Questo faceva sorgere dei problemi di natura combinatoriale,

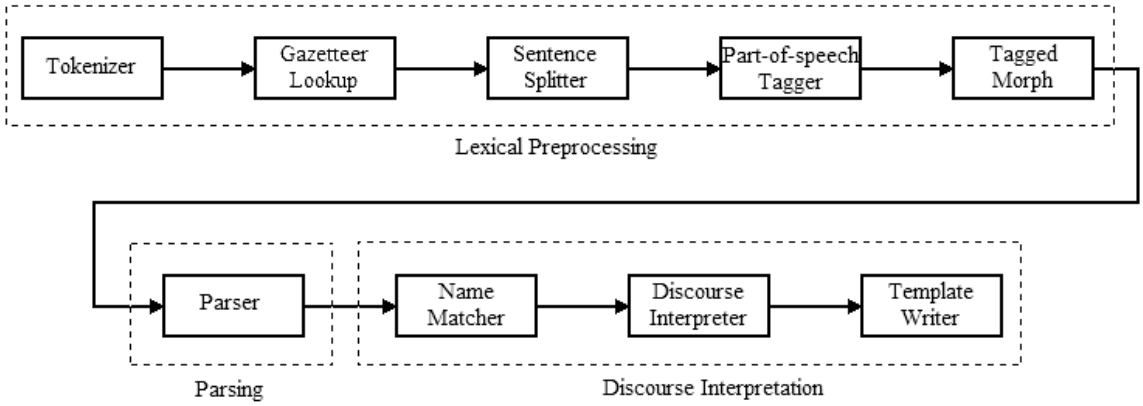


Figura 11 Architettura di LaSIE-II.

anche ottimizzando i parser tramite dati euristici o statistici, che comportavano lunghi tempi di elaborazione a causa della bassa capacità di calcolo disponibile nei primi anni ‘90.

Queste limitazioni hanno portato i ricercatori a cambiare approccio, sviluppando delle architetture alternative per i sistemi di Information Extraction basate su una combinazione di metodi statistici e tecniche di *shallow analysis*, ossia analisi più superficiali di quelle descritte in precedenza che usano soprattutto parser a stati finiti. Queste procedure possono comunque essere affiancate da tecniche di analisi linguistica nei casi in cui si siano dimostrate utili e affidabili.

In seguito viene illustrata una possibile architettura per un programma di Information Extraction. Tra i vari applicativi disponibili sul mercato si era scelto di basarsi su ANNIE, il *tool* utilizzato per compiere Information Extraction all'interno di GATE (General Architecture for Text Engineering), una suite freeware sviluppata dall’Università di Sheffield che consente di svolgere diverse operazioni nell'ambito del Natural Language Processing (presentato in [13], ultima versione rilasciata nel Novembre 2012). Questo sistema deriva a sua volta da LaSIE-II (Large Scale Information Extraction) [14], uno dei migliori software nei punteggi del MUC-7. La sua architettura è stata quindi scelta come base per questa esposizione per la maggiore modularità che la caratterizza rispetto a quella di ANNIE. Si ritiene, infatti, che con questa scelta sia possibile svolgere un’esposizione più approfondita del problema, essendo comunque le due strutture equivalenti e con diversi moduli in comune. Esempi di architetture alternative usate da partecipanti in varie edizioni del MUC come PROTEUS e SIFT si possono trovare in [15].

6.2.1 Panoramica dell'architettura

Come si può vedere in Figura 11, l'architettura proposta è altamente modularizzata e basata su quella utilizzata in LaSIE-II [14]. Pur non essendo di concezione molto recente l’organizzazione proposta per questo sistema è ancora valida e utilizzata [9] [13] [15]; la descrizione qui proposta fa quindi uso anche di osservazioni più recenti. In questa architettura

ogni modulo processa l'intero testo prima che venga invocato il successivo. Diamo ora una breve descrizione di ogni componente:

- Il *Tokenizer* identifica le zone del documento che saranno escluse dall'elaborazione (se, ad esempio, è presente una zona in cui sono salvate informazioni come l'origine del testo, la data di pubblicazione etc.) e divide la zona rimanente in *token*.
- Il *Gazetteer Lookup* cerca le corrispondenze tra gruppi di una o più parole del testo e gli elementi presenti all'interno di una serie di liste definite dall'utente, contenenti sia nomi specifici (luoghi geografici, organizzazioni, etc.) sia parole chiave (nomi propri, elementi delle date, etc.), etichettando i riscontri con la categoria di cui il gruppo fa parte.
- Il *Sentence Splitter* divide il testo in frasi distinte.
- Il *Part-of-speech Tagger* esegue un'analisi grammaticale del testo assegnando una etichetta ad ogni token.
- Il *Tagged Morph* esegue un'analisi morfologica per identificare la forma base dei token, nel caso in cui siano stati etichettati come nomi, verbi o aggettivi.
- Il Parser esegue un'analisi del testo con una o più grammatiche e termina con l'assegnazione alla frase della *best parse*, che può essere anche parziale.
- Il *Name Matcher* abbina le parole del testo che indicano la stessa entità.
- Il *Discourse Interpreter* aggiunge tutte le informazioni presupposte dall'input (se, ad esempio, è stata specificata una data di pubblicazione al testo in esame si può assegnare un valore preciso a parole come ieri e domani) e termina la Coreference Resolution iniziata dal Name Matcher.
- Il *Template Writer* inserisce le informazioni richieste dallo schema definito dall'utente.

Si nota come questa architettura non segua fedelmente la suddivisione delle attività illustrata nella Sezione 6.1, in quanto alcune di esse sono divise in più moduli o risultano accoppiate in uno unico, come è il caso della Template Element Construction, della Template Relation Construction e della Scenario Template Production che sono per la maggior parte unite nel Template Writer.

6.2.2 Lexical preprocessing

Il preprocessamento lessicale include i primi cinque moduli considerati in questa architettura. Il suo scopo è fornire al parser tutte le informazioni necessarie a ricostruire nel modo più efficiente e accurato possibile la struttura della frase.

Il funzionamento del Tokenizer e del Sentence Splitter è semplice e non richiede particolari commenti. L'analisi morfologica svolta nel Tagged Morph è anch'essa realizzata in modo semplice e utilizza sia un set di espressioni regolari sia una lista di tutte le forme irregolari. Entrambi devono essere determinati manualmente o in modo semi automatico, basandosi in ogni caso su analisi già compiute a riguardo sulla lingua in cui si opera.

Il Gazetteer Lookup tenta di identificare ed etichettare parole o locuzioni associate a entità specifiche, compiendo un confronto con una serie di liste contenenti sia nomi propri, luoghi geografici e simili sia nomi comuni che si comportano come indicatori affidabili di classi di entità, come accade per i titoli ("Presidente", "Giudice", etc.) o per le compagnie ("Associazione", "Compagnia", etc.). La creazione di queste liste avviene in due modi distinti: quelle più specifiche, ossia quelle del secondo tipo, devono essere compilate manualmente e sono spesso specifiche al dominio d'interesse mentre quelle del primo tipo sono più indipendenti dall'applicazione e si possono spesso creare a partire da liste già esistenti, come i risultati di un censimento o un atlante geografico. Questo modulo è posto prima del Sentence Splitter in modo che parole come "S.P.A." non causino una divisione sbagliata delle frasi.

Come Part-of-speech Tagger è usato normalmente quello sviluppato da Brill [16] poiché il suo funzionamento e la sua implementazione sono molto semplici e offre un'accuratezza equivalente ai sistemi più complessi basati completamente su metodi statistici. Questi ultimi compiono l'analisi grammaticale del testo assegnando i vari tag in base alla distribuzione di frequenza che hanno certe sequenze di tag, misurate analizzando un corpus di testi già annotati, e tramite regole ricavate appositamente per migliorare le prestazioni. Il tagger sviluppato da Brill, invece, assegna ad ogni parola presente nel suo dizionario la parte del discorso per lei più probabile (ad esempio supponendo che la parola "viaggio" sia al 60% un nome e al 40% un verbo assegnerà un tag corrispondente a nome) mentre identifica ogni parola sconosciuta o come un nome o come il tag più comune per il suffisso di quella parola (ad esempio una parola inesistente come "stratemare" sarebbe classificata come verbo perché termina con il suffisso -are). Il dizionario viene costruito compilando una lista del tag più comune per ogni parola presente nel corpus di addestramento. A queste annotazioni sono poi applicate un certo numero di *patch* che correggono le assegnazioni sbagliate. Le patch sono ricavate esaminando un cosiddetto *patch corpus* su cui viene prima eseguita una analisi con il tagger iniziale ottenuto fino quel momento per poi ricavare una serie di triple che indicano quante volte un tag è stato scambiato per un altro. Grazie a questi dati le patch sono create combinando delle strutture predefinite (ad esempio "cambia il tag A con il tag B se la parola precedente è di tipo C") con le triple ottenute.

La preparazione del tagger avviene in modo semi-automatico e non richiede la compilazione di regole particolari da parte dell'operatore. Essa richiede, però, che sia disponibile un corpus annotato sufficientemente grande per poter compiere le due operazioni di inizializzazione e di creazione delle patch. Il corpus, oltre a porre dei vincoli sulla lingua, è dipendente anche dal dominio d'interesse: si ottengono infatti prestazioni nettamente migliori usando un corpus dedicato sia al periodo storico che all'argomento che si sta analizzando. Si nota dunque come il corpus usato nel tagger e le liste di nomi usate dal Gazetteer Lookup dipendano in modo pesante dal campo d'interesse del ricercatore.

6.2.3 Parsing

Il parser compie l'analisi logica del testo in esame e assegna ad esse delle strutture semantiche utilizzando le informazioni ricavate nei moduli precedenti. Può essere organizzato in diversi modi: può essere diviso, infatti, in due parti, con la prima dedicata all'individuazione delle frasi nominali che costituiscono le entità d'interesse, portando quindi a termine la Named Entity Recognition, e la seconda che compie un'analisi generica, oppure può presentare solo la seconda funzionalità.

Le regole grammaticali su cui è basato il suo funzionamento possono essere ricavate in due modi diversi. Il primo metodo è semi automatico e si basa su un corpus annotato con i costrutti logici mentre il secondo è manuale e si fonda su principi generali o su regole ad hoc. Il secondo approccio è obbligato nel caso si voglia sviluppare un modulo separato per compiere la Named Entity Recognition, essendo le entità specifiche al problema e dovendo assegnare ad ognuna di esse una classificazione.

Lo sviluppo del parser generico si può compiere invece sfruttando entrambi gli approcci, ma in letteratura il secondo sembra essere quello più promettente. È possibile, infatti, facendo qualche semplificazione, ricavare una grammatica libera dal contesto da un corpus annotato ottenendo però un numero enorme di regole (17.540 regole da un corpus del Wall Street Journal [12]). Questo costringe ad eliminare le produzioni più rare, ottenendo dei risultati inferiori a quelli auspicati e dovendo inserire delle regole manualmente per migliorare le prestazioni. Per questo sembra preferibile un approccio totalmente manuale, basato sulla compilazione di un ristretto numero di regole a partire dai principi generali della grammatica della lingua in esame.

Alla fine dell'analisi, se più scomposizioni possono essere associate ad una stessa frase, viene selezionata euristicamente la best parse, che spesso può essere parziale, formata cioè da un insieme di alberi sintattici invece che da un albero unico. Non essendo a disposizione in questa fase informazioni sulla semantica è opportuno adottare un approccio il più possibile conservativo, assegnando solo quelle classificazioni che si è certi non aggiungano quasi mai errori; eventuali mancanze saranno corrette dal Discourse Interpreter che ha a disposizione queste informazioni.

A questo punto dell'elaborazione sono assegnate le strutture semantiche, le cui regole possono essere derivate in modo abbastanza affidabile da un corpus annotato: ad esempio per verbi e nomi la loro radice morfologica può essere usata come identificatore nella semantica, con il tempo e il numero inseriti come suoi attributi.

6.2.4 Interpretazione del discorso

In questo livello sono completate la maggior parte delle attività in cui è divisa l'Information Extraction: è in questo passo, infatti, che sono svolte la Coreference Resolution,

la Template Element Construction, la Template Relation Construction e la Scenario Template Production. Questo livello è inoltre il più dipendente dal dominio d'interesse perché i suoi risultati sono basati sugli schemi definiti dall'utente finale per l'esposizione dei risultati.

Dopo essere state analizzate dal parser, le frasi che compongono il testo in esame vengono analizzate sequenzialmente e la loro struttura semantica viene aggiunta ad un modello globale del testo che mira a rappresentare il discorso. Questo modello può essere rappresentato come un grafo, in cui ogni nodo rappresenta un concetto del discorso e gli attributi a lui associati mentre gli archi rappresentano le relazioni tra i vari concetti, stabilendo così una loro gerarchia. L'associazione delle strutture restituite in output dal parser al modello utilizzato dal Discourse Interpreter può essere migliorata usando una mappatura da parola a concetto generale, simile a quella che dovrebbe essere implementata per i codici aggregati descritti nella Sezione 3.1, anche se meno generica di questa.

Il modello generale utilizzato in questo punto deriva direttamente dalla definizione della struttura che deve essere compilata alla fine della procedura e sarà completato solo alla fine della Coreference Resolution, quando le varie istanze ipotizzate nella creazione saranno sostituite da quelle effettivamente menzionate nel testo. Ricollegandosi all'esempio della Sezione 6.1, la presenza nella frase della locuzione "è stato rapinato" comporterà la creazione del modello corrispondente ad un evento di rapina in cui sarà ipotizzata la presenza di istanze del tipo "rapinatore", "vittima", "luogo" e "tempo", che potrebbero però non essere direttamente compilabili con le informazioni restituite dal parser. Grazie a queste ipotesi si può inoltre risolvere la nominalizzazione dei verbi (nell'esempio di prima al posto di "è stato rapinato" poteva essere scritto "è stato vittima di una rapina", non avendo quindi un verbo come indicatore del tipo di evento) e completare il parsing se aveva restituito un'analisi incompleta non essendo riuscito ad individuare univocamente tutti i componenti della frase.

Dopo l'aggiunta al modello globale viene compiuta la Coreference Resolution. La risoluzione dei nomi propri è stata svolta in precedenza dal Name Matcher, basandosi spesso su un gruppo di regole ricavate manualmente. La risoluzione delle anafore invece è svolta con le informazioni disponibili in questo momento. Viene infatti compiuto un confronto tra ogni nuovo oggetto inserito nel modello e tutti quelli già presenti a partire dagli elementi interni alla frase per poi continuare sequenzialmente con le frasi precedenti. Ogni coppia, se riconosciuta come compatibile in quanto presenta attributi condivisi e un percorso diretto nel grafo tra gli elementi della coppia stessa, viene fusa a rappresentare un'unica entità nell'analisi. In questa fase è svolta anche la Template Element Construction per le frasi nominali, aggiungendo al soggetto le proprietà specificate dal predicato nominale della frase. È possibile inoltre programmare in questo punto la risoluzione di alcuni casi di catafora, ossia la relazione inversa all'anafora in cui il pronome precede l'elemento della frase a cui si riferisce.

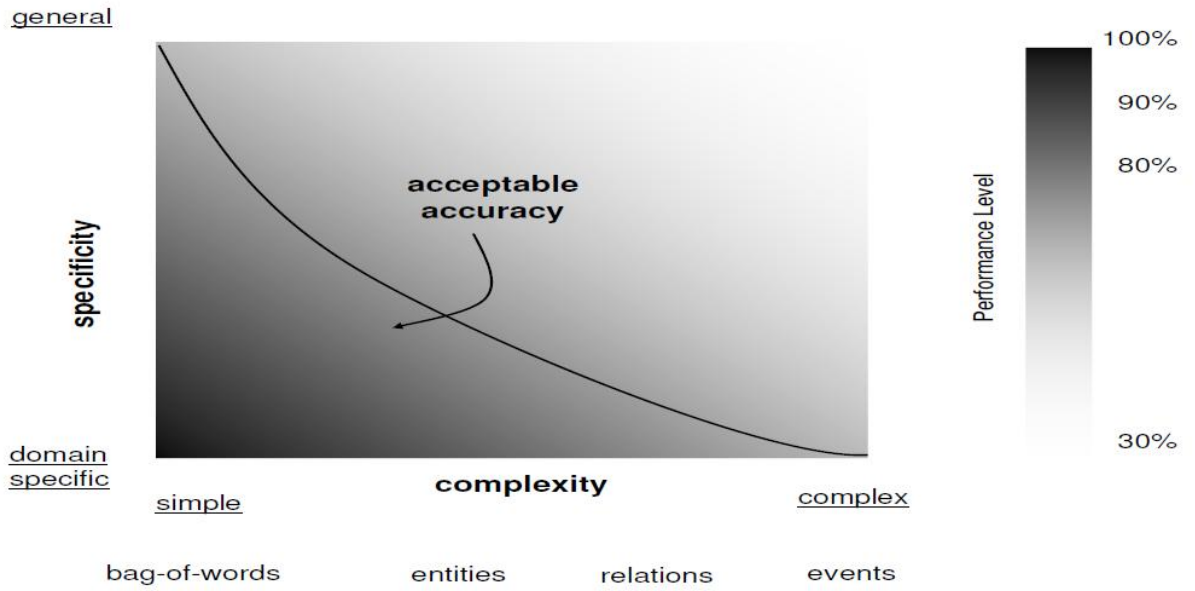


Figura 12 Grafico delle prestazioni di un sistema di Information Extraction rispetto alla complessità dei dati richiesti e alla specificità del dominio, tratto da [10].

Con i dati restituiti dal Discourse Interpreter il Template Writer ha a disposizione tutte le informazioni per poter compilare la struttura corrispondente all'evento descritto dal testo in esame.

6.3 Automatizzare l'acquisizione dei dati in PC-ACE

Si discute ora la possibilità di applicare l'Information Extraction alla procedura di acquisizione dei dati in PC-ACE. Prima di analizzare la sua compatibilità con le restrizioni imposte dalla Quantitative Narrative Analysis e la sua possibile integrazione in PC-ACE, che saranno affrontate nelle due Sottosezioni seguenti, bisogna illustrare due limiti generali di questa tecnologia.

Infatti, come si può vedere dal grafico in Figura 12, le sue prestazioni sono legate sia alla complessità delle informazioni che devono essere estratte, che varia a seconda che si vogliano ricavare singole parole e locuzioni o eventi completi, sia alla specificità del dominio di interesse, imponendo un trade-off tra le due. Più complessi sono i dati da estrarre più specifico deve essere il dominio del discorso mentre più semplici ed elementari sono le informazioni richieste più generale può essere il sistema utilizzato. Tra gli aspetti in cui può essere diviso il dominio d'interesse due sono i parametri che più influenzano l'affidabilità generale: il genere del testo analizzato e il soggetto dell'analisi. Entrambi stabiliscono, infatti, lo stile e il lessico impiegato e il livello di strutturazione proprio del documento. In generale è

molto più facile trarre informazioni da un documento semi strutturato come una cartella clinica rispetto che da un diario della degenza scritto dal paziente stesso.

Oltre che sull'affidabilità dell'operazione di estrazione, il dominio di interesse influisce anche sulla fase d'inizializzazione del processo. Infatti, gli schemi compilati nella Scenario Template Production sono specifici all'ambito di interesse nella ricerca, così come alcune delle liste usate nel Gazetteer Lookup ed alcune regole particolari che possono essere introdotte nella fase di interpretazione del discorso. Cambiare radicalmente oggetto della ricerca, ad esempio dai linciaggi avvenuti in Georgia all'analisi della propaganda nelle elezioni presidenziali, richiederebbe verosimilmente numerosi cambiamenti in vari livelli della procedura. Il superamento di questi vincoli usando tecniche di *machine learning* con l'obiettivo di ottenere sistemi per uso generale è tuttora oggetto di ricerca, come illustrato in [15].

I progetti di ricerca in cui viene applicata la Quantitative Narrative Analysis possono essere collocati nel quadrante superiore destro del grafico in Figura 12. I dati di interesse farebbero dunque parte di un dominio relativamente generico e avrebbero una complessità elevata: generalmente per loro non sarebbe garantita una grande affidabilità nell'estrazione, richiedendo in teoria numerose correzioni da parte del ricercatore. La flessibilità concessa da PC-ACE nella definizione della Story Grammar, invece, consente anche lo svolgimento di analisi non basate sulla struttura Soggetto-Azione-Oggetto dei testi narrativi ma anche su grammatiche più semplici e astratte. Per questi progetti l'affidabilità della procedura di Information Extraction sarebbe probabilmente più elevata, in quanto la struttura della Story Grammar utilizzata si avvicinerebbe molto di più ai template compilati all'estrazione dei dati e la complessità richiesta sarebbe minore.

6.3.1 L'Information Extraction e la Quantitative Narrative Analysis

La struttura di un template compilato nell'ultima fase dell'Information Extraction e di una tripletta semantica usata in una Story Grammar sono a prima vista molto simili: entrambi gli schemi, infatti, si basano su una struttura Soggetto-Azione-Oggetto con diversi modificatori assegnati ai vari elementi (vedere la Figura 13; l'esempio è tratto da un ipotetico progetto di ricerca riguardante episodi di criminalità nel mondo aziendale).

Si nota però una differenza fondamentale tra le due: la struttura della Story Grammar, infatti, si basa per definizione su un modello invariante presente nei testi di natura narrativa che sono soggetti alla Quantitative Narrative Analysis. Al contrario, i template usati nella Scenario Template Production sono specifici non solo al dominio d'interesse (questo costituisce un aspetto comune con le Story Grammar, poiché anche i loro modificatori possono cambiare da progetto a progetto), ma variano anche al suo interno: estraendo ad esempio dati riguardanti episodi di criminalità sarebbero usati schemi diversi per furti, omicidi, spaccio di droga, etc. oltre che per azioni più generiche quali i movimenti. Questa

24 Aprile 2013, Padova - Il locale comando dei Carabinieri ha annunciato l'arresto di Irene Tiepoli, amministratrice della Veneto Import-Export, dopo l'accusa di traffico di stupefacenti. Sarà sostituita nel suo ruolo da Ettore Del Corso.

(a)

24 Aprile 2013
id: ENT-001
tipo: data
valore: 24/04/2013

Padova
id: ENT-002
tipo: luogo
sottotipo: città

Carabinieri
id: ENT-003
tipo: organizzazione
luogo: ENT-002

Irene Tiepoli
id: ENT-004
tipo: persona
lavoro: manager
azienda: ENT-005

Veneto Import-Export
id: ENT-005
tipo: azienda
settore: import export

Ettore del Corso
id: ENT-006
tipo: persona
lavoro: sconosciuto
azienda: ENT-005

arresto
id: EVT-001
causa: traffico di droga
tempo: ENT-001
luogo: ENT-002
agente: ENT-003
vittima: ENT-004

successione
id: EVT-002
azienda: ENT-005
uscente: ENT-004
entrante: ENT-006

(b)

<+evento>
<+tripletta semantica>
<+soggetto>
<+organizzazione>
<organizzazione statale> - carabinieri
<soggetto aggregato> - forze dell'ordine
<luogo> - padova
<+azione>
<+azione semplice>
<verbo> - hanno arrestato
<verbo aggregato> - arresto
<+circostanze>
<+tempo>
<data> - 24/04/2013
<+luogo>
<città> - Padova
<+ragione>
<ragione semplice> - traffico di stupefacenti
<+oggetto>
<+persona>
<nome> - Irene Tiepoli
<+caratteristiche personali>
<sesso> - donna
<+lavoro>
<ruolo> - manager
<azienda> - Veneto Import-Export
<+tripletta semantica>
<+soggetto>
<+persona>
<nome> - Ettore del Corso
<+caratteristiche personali>
<sesso> - uomo
<+azione>
<+azione semplice>
<verbo> - sostituirà
<verbo aggregato> - succedere
<+circostanze>
<+ragione>
<tripletta semantica> - tripletta precedente
<+oggetto>
<+persona>
<nome> - Irene Tiepoli

Figura 13 Confronto tra il risultato di una operazione di Information Extraction (a), divisa nella lista delle entità individuate e in quella degli eventi estratti, e della compilazione di una Story Grammar (b), rappresentata sotto forma di albero e senza mostrare gli elementi vuoti.

divisione è necessaria perché solo in questo modo si possono specificare quali sono gli elementi della frase richiesti per completare quel template.

Una possibile soluzione a questo problema che meriterebbe di essere testata potrebbe essere rappresentata dallo sfruttamento dei codici aggregati di cui si è già parlato nella Sezione 3.1. È possibile infatti definire dei concetti generali a cui ogni azione appartiene, permettendo così di comporre un certo numero di schemi per il sistema di Information Extraction e ricavare così sia le triplette semantiche che gli oggetti in cui possono essere aggregate. Questi codici dovrebbero ovviamente essere definiti in anticipo dal ricercatore, al

quale sarebbe dunque richiesta una conoscenza di come le descrizioni dei fatti trattati possano variare.

Una volta superata questa differenza bisognerebbe anche ideare una mappatura tra le categorie degli schemi prodotti dal processo di Information Extraction e gli elementi della Story Grammar associata al progetto. Questo passaggio sembrerebbe fattibile con uno sforzo relativamente contenuto in quanto non dovrebbe essere difficile realizzare una relazione univoca tra le due rappresentazioni potendo intervenire sulle definizioni di entrambe le strutture.

6.3.2 L'Information Extraction e PC-ACE

Dopo la risoluzione dei problemi esposti finora bisogna affrontare la difficoltà principale, ossia come integrare all'interno di PC-ACE una procedura che renda possibile l'estrazione automatica delle informazioni dai documenti.

La grande complessità della procedura stessa rende sconsigliabile una sua implementazione da zero. Si dovrebbe dunque sfruttare una libreria già esistente, limitando se possibile la scelta a quelle disponibili sotto forma di *Dynamic-Link Library* classiche, scritte dunque o in ambiente .NET o basandosi sulla API di Windows, e non a quelle disponibili in altri formati, come le librerie Java: non è infatti realizzabile un collegamento efficiente di queste ultime con il codice VBA sottostante a PC-ACE. Questa limitazione costringe ad eliminare dalle possibili scelte GATE [13], una delle suite disponibili con più funzionalità, così come NLKT (Natural Language Toolkit) [17], OpenNLP e UIMA [18]. Un altro limite deriva dalla natura open-source e freeware di PC-ACE e impone l'utilizzo di soluzioni non commerciali.

La necessità di integrare un tool per compiere Information Extraction in PC-ACE dovrebbe comunque essere sfruttata per realizzare una completa riscrittura del programma. Nonostante sia un'operazione molto lunga da intraprendere, soprattutto dopo aver raggiunto questo stadio dello sviluppo del programma, in questo modo si risolverebbero tutte le criticità rappresentate dalla scarsità di documentazione di una parte del codice e della dipendenza da un applicativo proprietario per il funzionamento. Il suo possibile utilizzo verrebbe inoltre esteso anche a ricercatori non in possesso di computer con sistema operativo Windows e si renderebbe possibile l'espansione del codice da parte di una comunità più grande di programmatori.

7 Conclusioni

Nello svolgimento di questa tesi si è affrontato il problema di come aumentare l'utilizzabilità di PC-ACE, un programma basato su Microsoft Access sviluppato per compiere Quantitative Narrative Analysis, una particolare tecnica di analisi dei testi narrativi usata nell'ambito delle scienze sociali. Si è intervenuti sulla sua procedura d'inserimento dei dati in quanto costituisce la fase più lunga di ogni progetto di ricerca in termini di tempo.

Si è agito su due livelli distinti. Dal punto di vista applicativo si è progettata una nuova interfaccia grafica per l'inserimento dei dati che consente la visualizzazione all'interno di PC-ACE del documento da cui si stanno estraendo le informazioni e che sfrutta pienamente la struttura ad albero propria delle Story Grammar, lo schema alla base della Quantitative Narrative Analysis in cui i dati sono inseriti. Essa è stata realizzata tramite l'integrazione in una maschera di Access di AdvancedTree e DocumentViewer, due controlli ActiveX realizzati appositamente per questa tesi, che in un futuro potrebbero essere estesi facilmente sia per compiere operazioni più complesse all'interno del programma sia per essere utilizzati in un ambiente esterno allo stesso.

L'obiettivo nel medio-lungo termine dei creatori di PC-ACE, comunque, è poter compiere in modo automatizzato la maggior parte delle operazioni non legate alla ricerca vera e propria, quali l'inserimento dei dati. A completamento dell'analisi si è eseguito uno studio teorico riguardante la possibilità di applicazione delle tecniche di Information Extraction alla procedura di acquisizione dei dati in PC-ACE. È stato proposto un esempio di una sua possibile architettura implementativa ed è stato evidenziato un insieme di problemi emersi durante questa trattazione che devono essere superati per poter integrare con successo questa tecnica all'interno del programma. Da questa analisi si è ricavato che una applicazione dell'Information Extraction alla procedura di inserimento dei dati merita di essere approfondita tramite dei test dedicati, ma un suo inserimento nel programma costringerebbe con tutta probabilità a riprogrammare PC-ACE, rendendolo indipendente da Microsoft Access.

8 Bibliografia

[1] Krippendorff Klaus, "**Content Analysis**", *International Encyclopedia of Communication*, vol.1, pp. 403-407, Oxford University Press, 1989.

[2] Stemler Steve, "**An overview of content analysis**", *Practical Assessment, Research and Evaluation*, vol. 7, no. 17, 2001. http://PAREonline.net/getvn.asp?v=7&n=17, consultato il 14/09/2013.

[3] Franzosi Roberto, "**Quantitative Narrative Analysis**", SAGE Pubblications, 2010.

[4] Franzosi Roberto, Doyle Sophie, McClelland Laura, Rankin Caddie Putnam e Vicari Stefania, ''**Quantitative Narrative Analysis Software Options Compared: PC-ACE and CAQDAS (ATLAS.ti, MAXqda, and NVivo)**'', *Quality and Quantity*, vol. 47, no. 6, pp. 3219-3247, 2013.

[5] Franzosi Roberto, De Fazio Gianluca e Vicari Stefania, "**Ways of Measuring Agency: An Application of Quantitative Narrative Analysis to Lynchings in Georgia (1875–1930)",** *Social Methodology*, vol. 42, no. 1, pp. 1-42, 2012.

[6] Franzosi Roberto, De Fazio Gianluca e Vicari Stefania, "**PC-ACE User's Manual**", (presente all'interno di PC-ACE v.828), 2009.

[7] AA.VV., "**MFC ActiveX Control**", Microsoft Developer Network, http://msdn.microsoft.com/en-us/library/k194shk8.aspx, consultato il 29/08/2013.

[8] AA.VV., "**Introduction to ActiveX Controls**", Microsoft Developer Network, http://msdn.microsoft.com/en-us/library/aa751972%28v=vs.85%29.aspx, consultato il 6/09/2013.

[9] Sarawagi Sunita, "**Information Extraction**", *Foundation and Trends in Databases*, vol.1, no. 3, pp. 261-377, 2008.

[10] Cunningham Hamish, "**Information Extraction, Automatic**", *Encyclopedia of Language and Linguistics, 2nd Edition*, Elsevier, 2005

[11] Appelt Douglas, "**An Introduction to Information Extraction**", *Artificial Intelligence Communications*, vol. 12, no. 3, pp. 161-172, 1999.

[12] Gaizauskas Robert, Wakao Takahiro, Humphreys Kevin, Cunningham Hamish e Wilks Yorick, "**Description of the LaSIE System as Used for MUC-6**", *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 207-220, Morgan Kaufmann, 1995.

[13] Cunningham Hamish, Maynard Diana, Bontcheva Kalina e Tablan Valentin, "**GATE: an Architecture for Development of Robust HLT Applications**", *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, pp. 168–175, 2002.

[14] Humphreys Kevin, Gaizauskas Robert, Azzam Saliha, Huyck Chris, Mitchell Brian e Cunningham Hamish, "**Description of the LaSIE-II System as Used for MUC-7**", *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998. http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html, consultato il 09/09/2013.

[15] Turmo Jordi, Ageno Alicia e Català Neus, "**Adaptive information extraction**", *ACM Computing Surveys*, vol. 38, no. 2, pp. 4-61, 2006.

[16] Brill Eric, "**A Simple Rule-based Part-of-speech Tagger**", *Proceedings of the Third Conference on Applied Natural Language Processing*, pp. 152-155, 1992.

[17] Bird Steven, "**NLTK: the natural language toolkit**", *Proceedings of the COLING/ACL on Interactive presentation sessions*, pp. 69–72, 2006.

[18] Ferrucci David e Lally Adam, "**UIMA: an architectural approach to unstructured information processing in the corporate research environment**", *Natural Language Engineering*, vol. 10, no. 3-4, pp. 327-348, Cambridge University Press, 2004.