Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica – *Nuovo Ordinamento*

Information Extraction: il rapporto GATE - Named Entity Recognition

Relatore:
Prof. Sonia Bergamaschi

Candidato:
Simone Ferrari

Anno Accademico 2005-2006

Parole chiave:
Language Engineering
Information Extraction
Named Entity Recognition
Annotazione

RINGRAZIAMENTI

Il primo, più sentito, più diretto ringraziamento va senza dubbio alla mia famiglia, che mi è sempre stata vicina e mi ha sempre dimostrato l'affetto di cui ho avuto bisogno durante tutto questo percorso.

Desidero ringraziare inoltre la Professoressa Sonia Bergamaschi per il costante supporto fornitomi durante lo svolgimento della tesi.

Come ultimo ringraziamento, desidero citare i miei compagni di corso, con i quali ci siamo fatti forza a vicenda nei momenti difficili di questi 3 anni caratterizzati da enormi fatiche ed immense soddisfazioni.

Sinceramente, GRAZIE a Voi tutti…

Simone Ferrari

INDICE

Capitolo 4: Applicazione sperimentale

INDICE DELLE FIGURE

Introduzione

Negli ultimi 20 anni una delle discipline della *computer science* che ha registrato lo sviluppo maggiore è stata senza dubbio la *language engineering*, la cui definizione secondo Cunningham è

…la disciplina o azione dei sistemi software dell'ingegneria che realizza attività coinvolgendo l'elaborazione del linguaggio umano. Sia il processo di costruzione che il suo output sono misurabili e predicibili.

I motivi di questo forte progredire sono associabili al continuo avanzamento delle tecniche relative al Web, che in questo lasso di tempo partendo da zero sono riuscite a creare un rapporto inossidabile con l'uomo. Il misurarsi quotidianamente con una infinità di testi completamente diversi l'uno dall'altro ha fatto emergere nell'uomo il bisogno di un aiuto da parte dei computer nello scremare e scegliere le informazioni di cui effettivamente ha bisogno. Da qui la nascita della language engineering e del NLP, *Natural Language Processing* (Elaborazione del linguaggio naturale), ovvero quel processo di estrazione di informazioni semantiche da espressioni del linguaggio umano, scritte o parlate, tramite l'elaborazione di un computer. Sono facilmente intuibili le difficoltà intrinseche nella realizzazione di questa attività, basti pensare ai fraintendimenti che possono accadere in un discorso tra due persone e ai significati ambigui o ai doppi sensi che molte parole di molte lingue presentano. Per far fronte a ciò, tale attività viene suddivisa in fasi diverse, tuttavia simili a quelle che si possono incontrare nel processo di elaborazione di un linguaggio di programmazione:

- *analisi lessicale*: la scomposizione di un'espressione linguistica in token (in questo caso parole)

- *analisi sintattica*: arrangiamento dei token in una struttura sintattica (ad albero: parse tree)

- *analisi semantica*: assegnazione di un significato alla struttura sintattica e, di conseguenza, all'espressione linguistica

Una delle sub-discipline della language engineering che ultimamente sta prendendo piede è *l'Information Extraction*: il suo scopo è quello di estrarre informazioni strutturate o semi-strutturate da documenti non strutturati leggibili dalle macchine.
L'IE cerca di applicare metodi e tecnologie provenienti dalla pratica computer science, come per esempio intelligenze artificiali, al problema di processare automaticamente dati testuali non strutturati, con l'obiettivo di estrarre della *conoscenza strutturata* in qualche dominio.
Un tipico esempio è l'estrazione delle informazioni riguardanti eventi di fusione tra compagnie, ovvero relazioni del tipo $MERGE(company_1,company_2,date)$ da news online di finanza.
Il concetto di IE nasce durante le *Message Understanding Conferences* (MUC), un progetto nel quale si condusse la ricerca NLP da un approccio puro, completo, general-purpose verso prototipi pratici, parziali e specifici di un dominio. Lo scopo di queste conferenze era la valutazione di sistemi per l'IE in termini di tre grandi classi: *progresso*,

adeguatezza, *diagnostica*. Brevemente descriviamo in che cosa consistono questi tre metodi di valutazione:

progresso: ci sono almeno tre modi di guardare al progresso: come una stima del corrente stato dell'arte, come una misura delle migliorie in relazione alle valutazioni precedenti, e come una misura delle migliorie in relazione al confronto con le performance dell'uomo sulla stessa attività. Per fare ciò bisogna garantire delle metriche di giudizio applicabili sia alle macchine che all'uomo, per fornire un modo utile di guardare a quanti dei dati attesi il sistema trova e a quali errori e di che tipo esso commette, e per offrire metodi di confronto delle performance di sistemi diversi.

adeguatezza: sebbene le attività di valutazioni imitino alcune attività della vita reale, sotto alcuni aspetti esse sono irrealistiche, per esempio la completa autonomia del processo di estrazione. Fin quando le attività rimangono costrette tra pareti come lo scopo della valutazione, non è possibile tradurre i risultati della valutazione direttamente in termini che riflettono i requisiti specifici di ogni particolare applicazione della vita reale, neanche applicazioni che hanno forti somiglianze con le attività di valutazione. Da qui, il bisogno di stimare l'adeguatezza delle metodologie di valutazione alle applicazioni della vita reale.

diagnostica: le metriche principali di recall e precision e quelle secondarie di undergeneration e overgeneration forniscono informazioni diagnostiche, nel senso che mostrano quanto è accurata la performance del sistema all'attuale livello di "copertura" dell'attività. Come punto di partenza per la valutazione delle prestazioni vengono usate tali metriche, ma durante la questa fase vengono effettuate delle prove utilizzando le attività di information extraction per rivelare la capacità di analizzare il testo. Queste valutazioni sono di tipi diagnostico, perché esse cercano di isolare specifici aspetti dell'analisi del testo dall'attività di information extraction, utilizzando testi di esempio selezionati dall'input dell'attività di estrazione.

Un riepilogo delle varie MUCs è riportato di seguito insieme agli argomenti di discussione:

- MUC-1 (1987), MUC-2 (1989): Messaggi riguardanti operazioni navali.
- MUC-3 (1991), MUC-4 (1992): Terrorismo nei paesi dell'America Latina.
- MUC-5 (1993): Joint ventures e microelettronica.
- MUC-6 (1995): Articoli di news sul management.
- MUC-7 (1998): Verbali di lanci di satelliti.

Questa tesi è indirizzata verso un'analisi tecnica dell'Information Extraction, fornendo la descrizione delle sue caratteristiche ed i possibili ambiti di impiego, nonché la sua definizione in cinque sottoattività, dedicando particolare attenzione ad una di queste, la Named Entity recognition. Viene presentato un sistema per la realizzazione di tale attività, GATE, descrivendo le risorse che utilizza, e viene valutato su alcuni siti web. L'organizzazione del documento è la seguente:

- nel capitolo 1 viene effettuata la distinzione (obbligatoria!) tra Information Retrievial ed Information Extraction, evidenziando i punti a favore ed a sfavore dell'una e dell'altra;

- il capitolo 2 è dedicato interamente all'Information Extraction, con cenni storici relativi alla sua nascita e sviluppo nell'ambito delle MUCs, con una descrizione dei caratteri che lo contraddistinguono e delle sue sottoattività, con una presentazione degli scenari futuri del suo impiego e gli sviluppi a venire;

- nel capitolo 3 il discorso si sposta su GATE, per il quale viene effettuata una lunga trattazione relativa alla sua organizzazione, al suo sviluppo, entrando poi nello specifico del suo modulo dedicato alla Named Entity recognition, ANNIE (A Nearly New IE system), del quale si analizzano le Processing Resources, ovvero quelle componenti che effettivamente processano i documenti in input e restituiscono in output le entità annotate. In ultimo, vengono presentati due strumenti che il sistema mette a disposizione per effettuare una valutazione delle prestazioni: l'AnnotationDiff Tool ed il Benchmarking Tool.

- il capitolo 4 invece è dedicato all'attività sperimentale, ovvero all'effettivo utilizzo di GATE per realizzare la Named Entity recognition. Come input per i sistema si utilizzano alcuni siti web alberghieri facenti parte del progetto WISDOM al quale ha collaborato la Professoressa Bergamaschi ed il suo DBGROUP. Grazie alle conoscenze acquisite nel capitolo 3, è possibile effettuare un'analisi delle operazioni compiute dalle Processing Resources sul documento, nonché valutare le prestazioni del sistema.

Capitolo 1
INFORMATION RETRIEVIAL vs INFORMATION EXTRACTION

1.1) Distinzione preliminare (start point)

Questo capitolo, più che capitolo 1, si sarebbe dovuto chiamare capitolo 0: è questo infatti il punto di partenza di tutta l'analisi seguente. Come si può notare, questo capitolo consta solo di un paragrafo: il motivo di questa scelta e di non averlo integrato nel capitolo successivo è il fatto di voler dargli tutta l'importanza e l'attenzione che merita: aver compreso il nucleo di questa distinzione significa trovarsi in sintonia con il resto della tesi.
E' importante effettuare una grande diversificazione tra queste 2 tecniche: un sistema di IR è molto spesso considerato un sistema che accetta come input un insieme di documenti (denominato "corpus") e una query (insieme di parole "chiave" rappresentative dell'argomento d'interesse) e restituisce solo quei documenti appartenenti al corpus che considera rilevanti in accordo con la query, cioè i documenti nei quali è più probabile che sia contenuto il dato specificato. Abitualmente, in aggiunta ai documenti in output, viene restituita per ognuno di essi una misura caratteristica della rilevanza (score): esistono diverse metriche per quantificare tale rilevanza, ma di sicuro la più immediata è quella di collocare i documenti più rilevanti come primi nell'output. A questo punto tocca all'utente leggere i documenti rilevanti ed estrarre da questi le informazioni cercate tramite la query.
Diverso è il comportamento di un sistema di IE: questa è una tecnologia basata sull'analisi del linguaggio naturale che permette l'estrazione di frammenti di informazioni. Questo processo presenta gli stessi input dell'IR ma restituisce in output dati poco ambigui in un formato prestabilito. Questi dati possono poi essere fatti visualizzare all'utente oppure immagazzinati in un database o foglio di calcolo per un'analisi successiva, o ancora usati come indici in applicazioni IR come un motore di ricerca su internet.
Dal confronto tra le 2 tecniche emergono lati positivi e negativi: l'IE è molto più difficile da implementare rispetto all'IR perché vincolato a vari gradi di analisi, e per alcune richieste è meno accurato rispetto alla lettura umana presente invece nell'IR; comunque quando l'analisi viene effettuata su una vasta quantità di documenti l'IE è più efficiente, perché permette di ridurre il tempo d'analisi eliminando il tempo della lettura di ricerca. Inoltre, quando i risultati hanno bisogno di essere rappresentati in diversi linguaggi, il formato prestabilito dell'output è molto più diretto rispetto alla traduzione.

Capitolo 2
DISCUSSIONE SU INFORMATION EXTRACTION[1]

2.1) Descrizione generale

L'IE nel senso discusso qui in precedenza nasce alla fine degli anni '80 – inizio anni '90 intorno alle Message Understanding Conferences (MUCs) (Grishman and Sundheim (1996); Sundheim (1995); Chincor (1998)). Questi eventi furono particolarmente distintivi perché impiegarono una precisa e diretta procedura di valutazione quantitativa dove diversi istituti di ricerca prima definivano una precisa attività, poi implementavano sistemi competitivi che venivano confrontati con i dati annotati dall'uomo. Sebbene questo tipo di procedura sperimentale sembra ovvia in molti altri rami della scienza, il suo utilizzo prima di MUC era insolito in molti contesti di *language processing* (probabilmente dovuto alle prime associazioni tra il lavoro e i campi fenomenologici come la linguistica, in opposizione alla sperimentazione empirica o all'ingegneria). Il ciclo sperimentale di MUC guidò verso grandi progressi nell'IE fino alla canonica definizione che illustreremo in seguito (sezione 2.4). Seguendo questo approccio, i lavori riguardanti l'IE presero parecchie direzioni, come la commercializzazione di alcune delle tecniche di base, lavori sullo sviluppo della portabilità e, più recentemente, la collaborazione al progetto generale del Semantic Web Berners-Lee(1999). Un importante traguardo degli ultimi 15 anni è che ora la gente comprende in modo ragionevolmente preciso i vari parametri che influenzano le performance della tecnologia nelle diverse impostazioni delle applicazioni (in altre parole il lavoro scientifico ha posato le basi per l'ingegneria dei sistemi reali). La sezione seguente illustra questi parametri.

2.2) Complessità vs Specificità

La differenza tra il progetto generale dell'analisi del linguaggio(o comprensione) e la più ristretta impresa dell'*extraction* è la risposta alle basse performance dell'analisi nel caso generale. Per capire meglio ciò, immaginiamo di voler realizzare un'applicazione che si serve dell'analisi del linguaggio come supporto a qualcuna delle sue funzioni. Supponiamo che le informazioni che possono essere estratte varino in *complessità* (es. siamo interessati semplicemente ai nome di persona o ad eventi complessi che coinvolgono molti partecipanti) e in *specificità* (es. siamo interessati ad informazioni generali riportate in qualsiasi modo in qualsiasi testo, o a settori specifici riportati in un certo modo in alcuni tipi di testo). Se noi trasferiamo su un grafico di coordinate complessità-specificità delle informazioni che verranno estratte i livelli di performance accettabile dei componenti dell'analisi (figure 1), notiamo che esiste un trade-off (compensazione) tra le due variabili: maggiore è la complessità dei dati che devono essere estratti, più specifica deve essere la sfera del discorso; più sono semplici i dati, più in generale gli algoritmi di estrazione possono essere applicati. Questa visione bidimensionale delle performance dell'IE nasconde una serie di delicati problemi. Per esempio, un testo che informa lo svolgimento di un seminario (testo molto specifico, molto semplice da analizzare per i sistemi IE) può essere più facile o difficile da processare a seconda del genere. Infatti, se il testo è riguardo una serie di seminari circa grandi uomini del passato, ci può essere ambiguità tra i nome di coloro che intervengono e quelli delle persone oggetto dei seminari. D'altro canto, se i seminari riguardano le telecomunicazioni mobili, la regolarità e la specificità della

[1] Hamish Cunningham, "Information Extraction, Automatic " pdf

terminologia tipica di questo ambito rende l'analisi strutturale più semplice. Perciò dobbiamo anche considerare i parametri *tipo di testo* e *oggetto*(dominio):
-tipo di testo: il tipo di testo con il quale stiamo lavorando, per esempio articolo di giornale, e-mail, romanzo, od output di un riconoscitore vocale.
-dominio: l'argomento oggetto dei testi, per esempio eventi mondiali, avvisi di seminari, riviste finanziarie, richieste di supporto tecnico, informazioni turistiche, e lo stile col quale sono scritti, per esempio formale o informale. Alla fine c'è il problema di a quali informazioni l'utente di IE è interessato: nomi di persona, lanci di missili, fusioni tra compagnie, problemi riscontrati con un particolare pacchetto software, o la descrizione di come localizzare una zona della città. Per questo si rimanda al paragrafo 2.4.

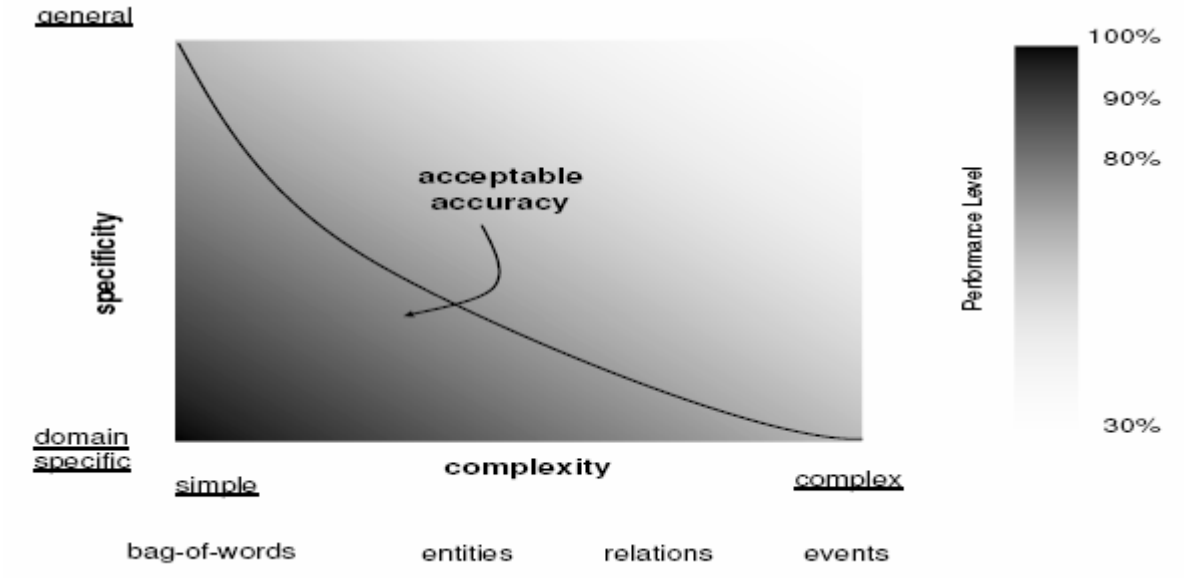


Figura 1. Trade-off della performance in relazione alla complessità e alla specificità

2.3) Ambiti di impiego (scenarios)

Questa sezione contiene alcuni scenari che illustrano come un software di applicazioni IE possa mediare tra il testo e i bisogni di informazioni strutturate di vari tipi di users. In ogni caso 'utente specifica il suo bisogno di informazione alla staff di sviluppo IE, il quale prepara un *extraction system* personalizzato per il cliente. Il materiale di input è definito come sottosezioni dl World Wide Web, o come materiale formale di notizie, siti di compagnie ecc, o come Web informale (weblog, mailing list). Gli sviluppatori devono analizzare il problema in accordo alle dimensioni evidenziate nella sezione precedente, e determinare come combinare "l'elaborazione" umana e della macchina così da conseguire il profilo di performance richiesto (see section 2.5.1).
I tipi di scenari descritti in precedenza sono in continuo aumento di rilevanza, dovuta in larga parte alla crescita esplosiva, alla natura dinamica e ai contenuti multilinguistici del Web.
- Analisi finanziaria
Il Web contiene numerose indicazioni di come una compagnia è vista, e quello o no che ci si aspetta che attui fortemente nel breve periodo. Alcuni di questi dati sono già altamente analizzati, come per esempio la notizie finanziarie scritte in inglese. Altri dati, scritti in

altre lingue o provenienti da sorgenti esaminate meno bene, sono voluminosi ed oscuri. L'IE può permettere all'analisi di rispondere a domande del tipo
"Quante istanze che predicono una forte performance ci sono per una determinata compagnia?"
"Negli anni passati come è cambiato il profilo di predizione per questa compagnia?"
"Quanti giudizi positivi/negativi sono stati espressi nei confronti della compagnia?"

- Strategia di marketing

La correzione dinamica agli investimenti di marketing è resa sempre più difficile dall'insufficienza di metriche che indichino l'impatto degli elementi della campagna. L'IE può sostenere la campagna di oggi basandosi sui risultati di ieri, producendo un output del tipo
"Il 7% degli articoli della stampa di oggi relativa all'Information Technology parla della sua compagnia. La proporzione media degli articoli direttamente riferiti alla sua compagnia è il 33%. Le citazioni per gli altri concorrenti chiave nel suo settore sono riassunte nella seguente tabella…..".
In modo molto simile, possiamo misurare il volume delle citazioni dovute ad eventi di pubblicizzazione:
"La compagnia Y espone a Modena. Nella settimana seguente l'esposizione il 20% della stampa che copre Modena ha menzionato Y".

- Lavori di Public Relations

Lo staff di Public Relations è interessato ad individuare il più in fretta possibile gli eventi che comportino un feedback negativo in modo tale da rispondere. L'IE può essere configurato per restituire il seguente tipo di output:
"La tabella seguente riassume 12 citazioni negative riguardanti la compagnia Y nelle ultime 24 ore delle notizie italiane…"

2.4) Cinque tipi di IE

Nel 1998, anno in cui terminò il progetto del MUC[2], si giunse alla definizione di IE suddivisa in cinque attività:

- **Named Entity recognition (NE)**
  trova e classifica nomi, posti, organizzazioni, ecc;
- **Coreference resolution (CO)**
  identifica le relazioni d'identità tra entità;
- **Template Element construction (TE)**
  aggiunge informazioni aggiuntive ai risultati del NE (usando CO);
- **Template Relation construction (TR)**
  trova relazioni tra entità TE;
- **Scenario Template production (ST)**
  trasferisce i risultati del TE e TR in uno specifico scenario d'evento.

In parole povere: NE si occupa della ricerca delle entità; CO comprende quali entità e riferimenti (come per esempio un pronome) indicano la stessa cosa; TE risale a quali attributi un'entità possiede; TR trova le relazioni tra entità; ST ricerca gli eventi ai quali le entità hanno partecipato.

[2] L'ultima conferenza fu MUC-7 SAIC (1998)

Consideriamo questa frase:

The shiny red rocket was fired on Tuesday. It is the brainchild of Dr. Big Head. Dr. Head is a staff scientist at We Build Rockets Inc.

Il NE scopre che le entità in gioco sono *rocket*, *Tuesday*, *Dr. Head* e *We Build Rockets Inc*. CO individua che *it* si riferisce a rocket. TE comprende che il rocket è *shiny red* ed è una *brainchild* del Dr. Head. TR scopre che il *Dr. Head* lavora per la *We Build Rockets Inc*. ST capisce che c'è stato un evento di lancio nel quale le varie entità sono state coinvolte.
Gli scenari presentati nella sezione precedente utilizzano questi cinque tipi di informazioni in varie combinazioni (e con vari profili tipici di performance).
Di seguito viene data una descrizione più dettagliata dei cinque tipi di IE.

2.4.1 Named Entity recognition (NE)

La tecnologia IE più semplice e più affidabile è *Named Entity recognition*. I sistemi NE identificano tutti i nomi di persona, i posti, le organizzazioni, le date, le quantità di soldi ecc. Nonostante le categorie del NE siano predefinite, ci sono varie opinioni su quali categorie debbano essere viste come *named entities* e quanto ampie queste categorie debbano essere. Alcune convenzioni emersero, e le entità sono comunemente evidenziate in accordo con il formato XML descritto nelle MUC. I tag "ENAMEX" sono usati per i nomi; "NUMEX" per le entità numeriche; "TIMEX" per le entità temporali.
Per esempio la frase *Jim bought 300 shares of Acme Corp. in 2006* viene analizzata come illustrato in figura 2 da un sistema IE, mentre il corrispondente codice XML è

<ENAMEX TYPE="PERSON">**Jim**</ENAMEX> bought <NUMEX TYPE="QUANTITY">**300**</NUMEX> shares of <ENAMEX TYPE="ORGANIZATION">**Acme Corp.**</ENAMEX> in <TIMEX TYPE="DATE">**2006**</TIMEX>.

Le categorie di base sulle quali si è convenuto includono le seguenti:

1) Names (enamex)
    - Organization
    - Person
    - Location

2) Times (timex)
    - Date
    - Time

3) Numbers (numex)
    - Money
    - Percent

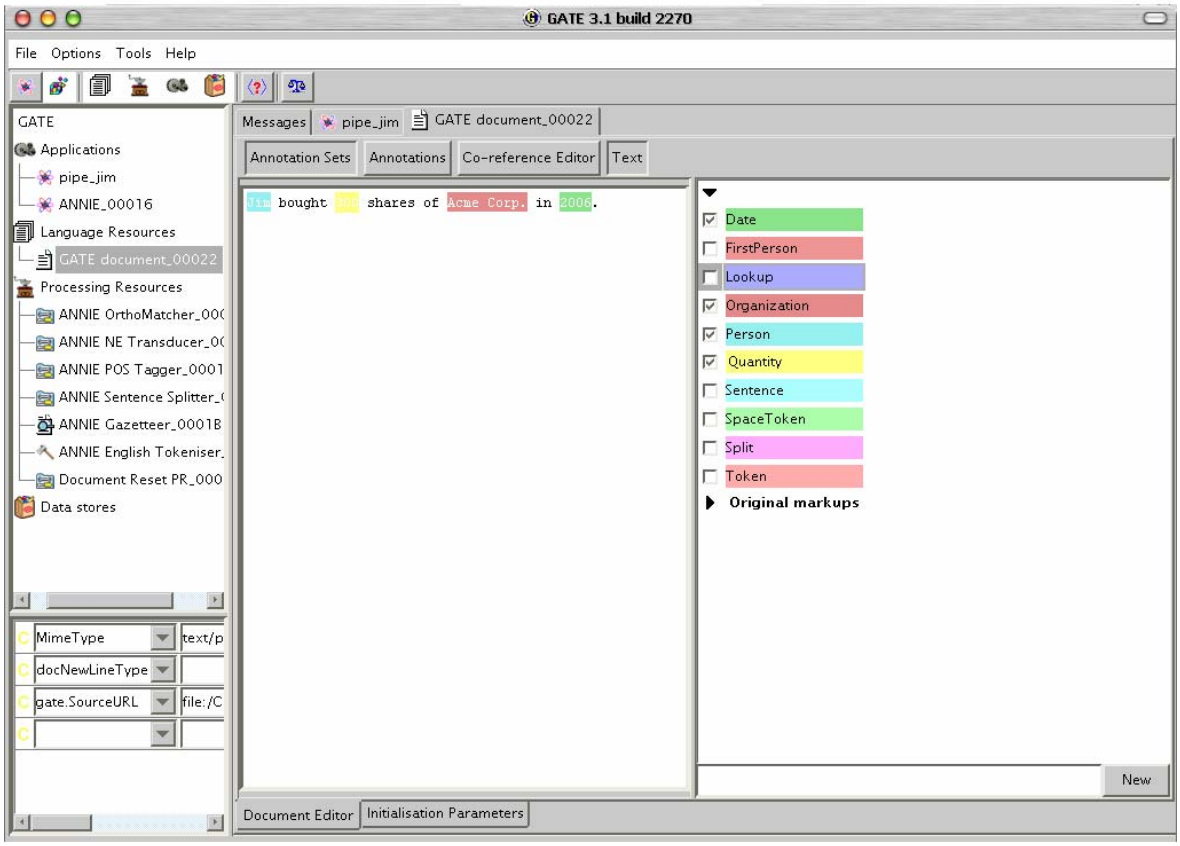


Figura 2. Snapshot di GATE

Comunque anche quelle che seguono possono essere considerate come categorie/subcategorie:

- Distance
- Speed
- Age
- Weight
- City
- Country
- State/Province
- River

Le categorie scelte per un particolare progetto NER possono dipendere dai requisiti del progetto. Se in un determinato campo è importante la classificazione numerica, le categorie collegate a dati numerici dovranno essere rifinite. In modo del tutto simile, se è importante una classificazione geografica, potrebbe essere necessario classificare ogni entità della categoria Location come particolare tipo di località.
Ma comunque ci sono anche delle difficoltà. La Named entity recognition, sebbene sembri un'attività semplice, mette di fronte ad una serie di sfide. In primo luogo, le entità possono essere difficili da trovare e , una volta trovate, difficili da classificare. Luoghi e nomi di persona possono essere gli stessi, e seguire una formattazione simile.

2.4.2 Coreference resolution (CO)

La Coreference resolution comporta l'identificazione delle relazioni d'identità tra le entità presenti nel testo. Queste entità possono essere quelle identificate dal NE recognition e riferimenti anaforici a quelle entità. Per esempio, nella frase

Alas, poor Yorick, I knew him Horatio

la CO collegherebbe "Yorick" con "him" (e "I" con Hamlet, se sono presenti informazioni sufficienti nel testo circostante).
Questo processo è meno rilevante all'utente rispetto alle altre attività dell'IE (siccome le altre attività producono un output che è di ovvia utilità per l'utente dell'applicazione, questa è più rilevante per i bisogni dello sviluppatore dell'applicazione). Possiamo usare CO per evidenziare in un testo tutte le occorrenze di uno stesso oggetto o per creare link ipertestuali tra esse; link che possono anche essere creati tra documenti. Ma comunque il significato principale di questa attività è di essere da appoggio per TE e ST (leggere successivamente). CO rende disponibile l'associazione di informazioni descrittive sparse per tutto il testo con le entità alle quali si riferisce. Per continuare il comune esempio di Shakespeare, coreference resolution permette all'analisi TE o TR di collocare Yorick in Danimarca. CO si può dividere in due sotto-problemi:
la risoluzione anaforica (es "I" con Hamlet); la risoluzione dei nomi propri. La "coreference identification" dei nomi propri trova le occorrenze dello stesso oggetto rappresentate con diversa ortografia e scrittura, come "IBM", "IBM Europe", "International Businnes Machines Ltd",…La risoluzione CO è un processo impreciso, specialmente quando applicata alla soluzione di referenze anaforiche. I risultati della CO variano largamente: a seconda del dominio probabilmente solo il 50%-60% può essere considerato affidabile. I sistemi CO sono dominio-dipendenti.

2.4.3 Template Element construction (TE)

Le attività TE vengono costruite su NE recognition e coreference resolution, associando informazioni descrittive con le entità. Per esempio, dal testo di figura 3 il sistema trova che "Bush administration" viene anche riportato come "government officials", e questo viene aggiunto come un alias. L'output del TE per il testo d'esempio è riportato nella figura 4; il formato è arbitrario. La cosa da notare è che questo è essenzialmente un database record, e potrebbe benissimo essere formattato per operazioni di store in SQL[3], o inserito in un foglio di calcolo elettronico, o (con qualche processo aggiuntivo) reso disponibile per una presentazione multilingue.

[3] Structured Query Language – un modo comune per manipolare i database relazionali

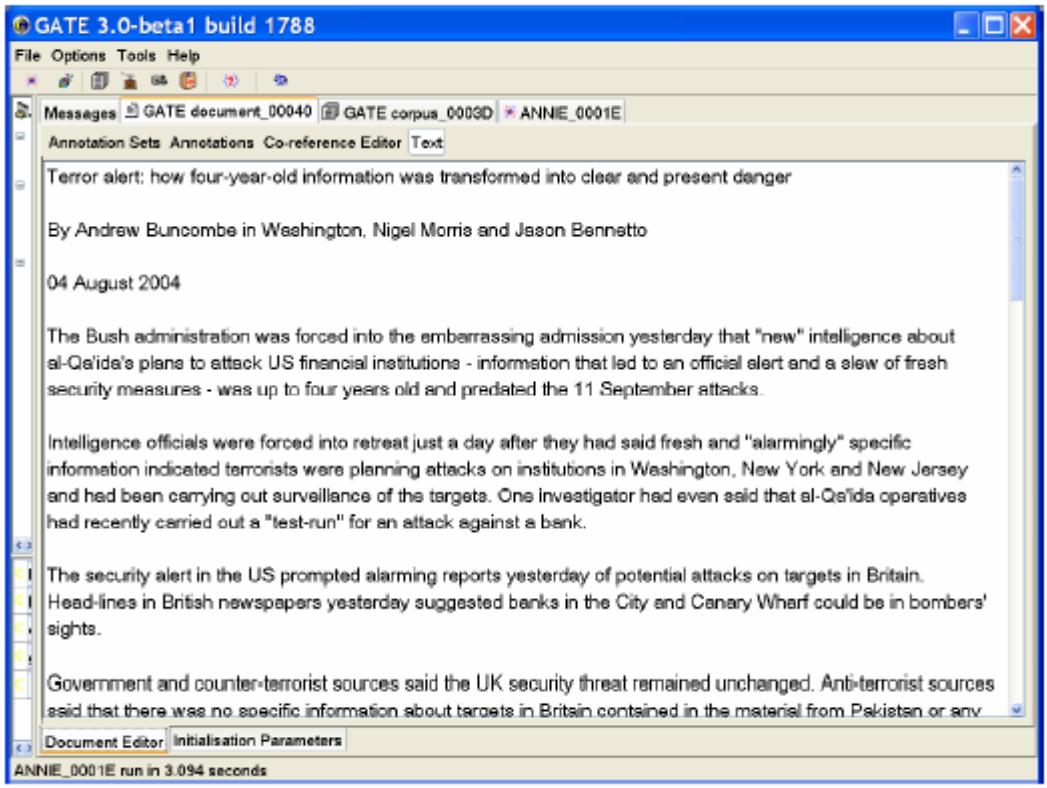


Figura 3. Testo di esempio

Punteggi buoni per sistemi TE sono intorno all'80% (per attività simili l'uomo può ottenere un punteggio del 95%). Come nel NE recognition, anche i risultati del TE sono debolmente dipendenti dal dominio, per esempio cambiare l'argomento oggetto del testo da notizie finanziarie ad altri tipi di notizie comporta qualche cambiamento al sistema, e cambiare da notizie a riviste scientifiche comporta cambiamenti abbastanza ampi.

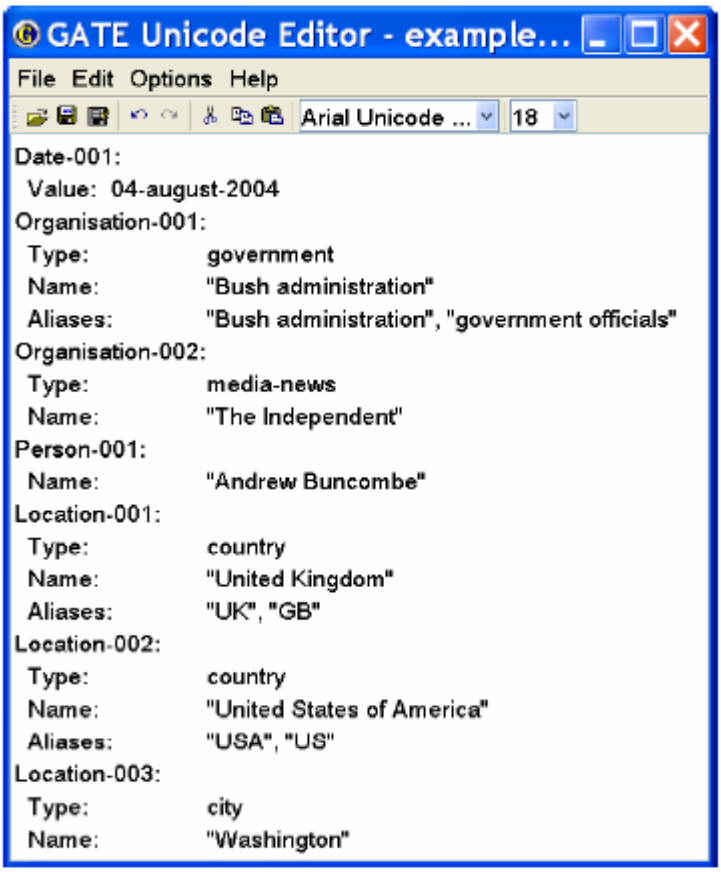


Figura 4. Template Elements

2.4.4 Template Relation construction (TR)

Prima del MUC-7, le relazioni tra entità erano parte del template dell'output delle valutazioni IE. Per permettere di "catturare" in modo largamente più vantaggioso le relazioni, MUC-7 introdusse l'attività TR –figura 5. Come descritto in Applet (1999), "l'attività di template relation richiede l'identificazione di un piccolo numero di possibili relazioni tra i template elements individuati nell'attività template element. Queste potrebbero essere il rapporto di lavoro tra una persona e una compagnia, una relazione di parentela tra due persone, una rapporto accessorio tra due compagnie. L'estrazione di relazioni tra entità è una caratteristica centrale di quasi ogni attività di information extraction, sebbene le possibilità nelle attività di estrazione nel mondo reale siano senza fine". In generale dei buoni punteggi per TR si raggiungono intorno al 75%. TR è un'attività debolmente dipendente dal dominio.

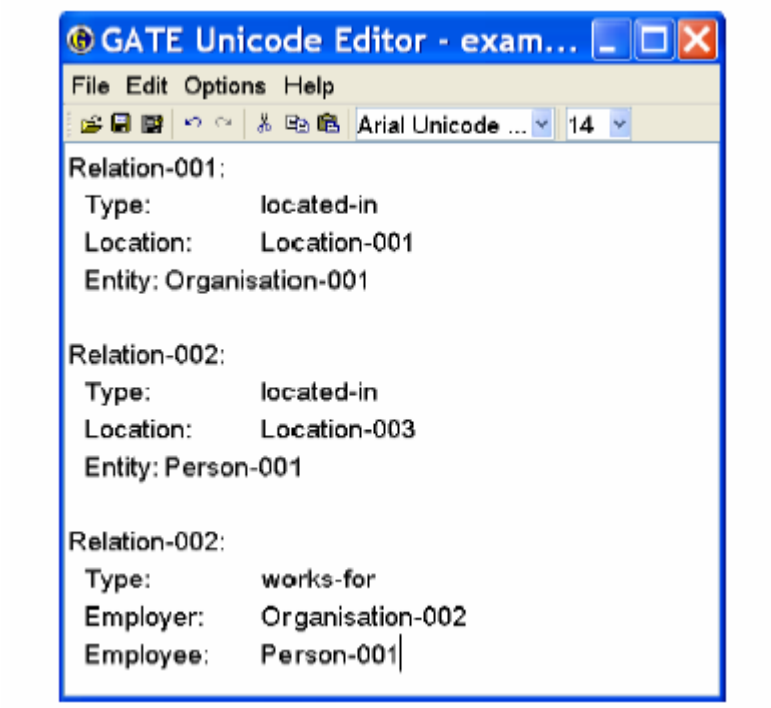


Figura 5. Template Relations

2.4.5 Scenario Template production (ST)

Scenario templates (STs) sono l'output prototipo dei sistemi IE, essendo l'attività originale per la quale il termine fu coniato. Essi collegano insieme entità TE e relazioni TR creando delle descrizioni di evento. Per esempio, TE può aver identificato Isabelle, Dominique e Françoise come entità persona presenti nell'edizione Robert delle lettere d'amore di Napoleone. ST può in seguito identificare avvenimenti come quello che Isabelle si trasferì a Parigi da Lione nel 1802 per stare più vicina al suo "piccolo ragazzo", che Dominique diede alle fiamme gli appartamenti di Isabelle, che Françoise scappò via con un antenato di Gerard Depardieu. Un esempio un po' più pertinente è presentato in figura 6, adattato da MUC-6 ARPA (1995).

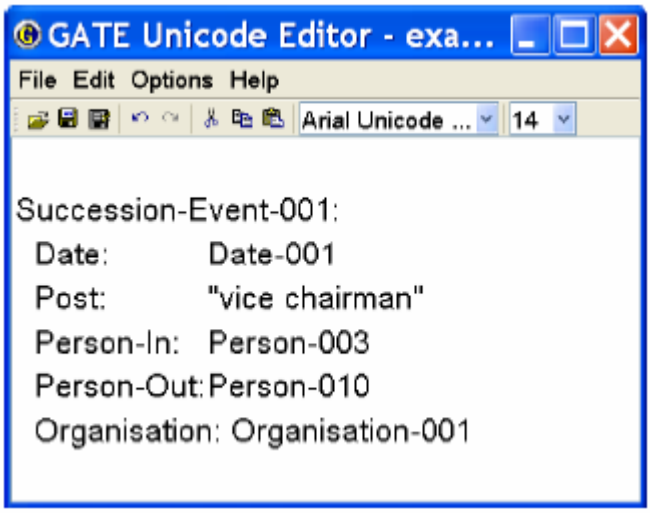


Figura 6. Scenario Template

ST è un'attività IE difficile; i migliori sistemi MUC raggiungono prestazioni intorno al 60%. Il punteggio "umano" può aggirarsi intorno all'80%, che illustra la complessità implicita. Questi aspetti devono essere tenuti in considerazione quando si prendono in esame appropriate applicazioni in tecnologia ST. Da notare, comunque, che si può aumentare la precisione al prezzo di effettuare più chiamate: possiamo sviluppare sistemi ST che non commettono molti errori, ma che tralasciano molte occorrenze di scenari rilevanti. In alternativa possiamo aumentare il numero delle chiamate e diminuire le perdite di dati, ma al prezzo di commettere più errori. L'attività ST è sia dipendente dal dominio sia, dalla definizione, collegata agli scenari di interesse degli utenti. Da notare che i risultati di NE, TR e TE vengono incamerati nel ST. Si noti in aggiunta che in MUC-6 e MUC-7 agli sviluppatori furono date le specifiche per l'attività ST solo un mese prima che il sistema venisse analizzato in prestazioni. Questo perché non ci si accorse che un sistema IE che necessitasse di molte e lunghe revisioni per far fronte a nuovi scenari valesse meno rispetto ad uno che potesse andare incontro a nuove specifiche abbastanza rapidamente (see section about portable IE). Da ciò si può trarre che i punteggi in MUC-6 e MUC-7 furono probabilmente leggermente inferiori a quelli che sarebbero stati con un periodo di sviluppo maggiore. L'esperienza dalle precedenti MUCs e dai lavori successivi suggerisce, comunque, che la tecnologia contemporanea ha difficoltà ad ottenere punteggi superiori al 60% in accuratezza per questa attività.

2.5) Extraction multilingua

I risultati descritti prima possono essere tradotti per presentarli all'utente o per essere immagazzinati in un database esistente. In generale questa attività è più facile rispetto alla traduzione di un testo consueto, ed è prossimo alla localizzazione del software, il procedimento del creare messaggi e label sui menu e sui pulsanti multilingua di un programma. La localizzazione avviene salvando alcune liste delle traduzioni dirette degli elementi conosciuti. Nel nostro caso dovremmo avere nelle liste le traduzioni per parole come "entity", "location", "date". Abbiamo anche bisogno di poter visualizzare data e numeri nei formati locali, ma per questo problema ci vengono incontro alcune librerie di codice. I problemi possono sorgere quando stralci arbitrari di testo vengono usati nelle strutture di descrizione delle entità, per esempio lo slot descriptor negli oggetti MUC-6 TE. In questo caso una frase fatta viene estratta dal testo, con qualunque aggettivo, proposizione relativa, ecc. che presenta all'interno, così il linguaggio è completamente senza restrizioni e necessita di un meccanismo di traduzione completa.

2.6) IE dopo MUC

In questa sezione viene riportata la storia aggiornata ad oggi, guardando agli sviluppi nella portabilità dei sistemi IE, al programma ACE che seguì MUC, e al progetto di annotazioni automatiche per applicazioni di Semantic Web.

2.6.1 IE portabile

Una particolare applicazione IE può essere configurata per processare articoli di notizie finanziarie provenienti da un particolare news provider (scritte nel linguaggio locale) e

trovare informazioni circa fusioni tra compagnie e vari altri scenari. Le performance del sistema possono essere predette solo per questo insieme di fattori. Se la richiesta fosse di estrarre avvenimenti dalle lettere d'amore di Napoleone Bonaparte come pubblicato in un bando del Comune di Parigi nel 1871, il livello di performance non sarebbe più predicibile. Adeguare un sistema IE a nuovi requisiti è un'attività che varia proporzionalmente al grado di variazione dei parametri discussi nella sezione 2.2.

Perciò una pista centrale della ricerca IE conduce al problema della *portabilità*, e questa è stata una delle aree di sviluppo più fertili dalla fine del programma MUC negli ultimi anni '90. Possiamo distinguere tre ampie sezioni di lavoro in questa area:

1) Imparare le regole dell'extraction di modelli da esempi annotati;
2) Inserire sistemi di studio all'interno dei sistemi end-user;
3) Supportare lo sviluppo di regole/modelli selezionando staff specializzati.

Il primo approccio è di imparare sezioni o tutto riguardo i sistemi di extraction attraverso "annotazioni d'addestramento"(annoted training data). Il vantaggio consiste in una riduzione del bisogno di staff specializzati per cercare di realizzare la portabilità del sistema. Gli svantaggi sono:

- possono essere estratti solo dati semplici, o dati complessi da testi semplici, come avvisi di seminari (infatti molti degli algoritmi attuali comuni per quest'area furono sviluppati per lo screen scraping[4], che è un'attività più semplice rispetto alla maggior parte delle analisi del linguaggio);
- è richiesta una grande quantità di training data.

Valutazioni sui lavori di quest'area possono essere trovate in Cardie (1997); Daelemans e Osborne (2003)
Nel secondo approccio, i ricercatori hanno provato ad inserire lo apprendimento dentro strumenti end-user dove gli utenti correggono i suggerimenti IE. Questo approccio sposta il problema dal costo di produzione dei training data tipico dell'approccio dello apprendimento all'accelerare il processo di annotazione. Un metodo ciclico conosciuto come conoscenza *mixed-initiative* è usato, dove l'utente inizia col fare manualmente tutti i lavori di annotazione mentre il sistema impara sullo sfondo. Quando la qualità dei sistemi studiati è alta abbastanza, il sistema può proporre le annotazioni all'utente; la correzione degli errori è riportata nell'algoritmo di apprendimento. Riferimenti in Day et al. (1997); Grishman (2001). La conoscenza mixed-initiative è stata ribattezzata *conoscenza attiva* nei lavori successive.
Nel terzo approccio, la portabilità dei sistemi IE può essere massimizzata creando un ambiente di sviluppo per uno staff specializzato per adattare un sistema base. I vantaggi sono:

- il sistema base può essere disegnato verso la robustezza e la portabilità;
- la complessità dei dati estratti non è limitata da un algoritmo di studio;
- tutti gli aspetti ingegneristici del processo possono essere presi in cura dell'infrastruttura (dalla visualizzazione dei dati alla valutazione delle performance).

[4] Screen scraping è il processo di cancellazione degli elementi di presentazione di dati o testi, es come visualizzarli in una pagina Web, in modo tale da consentire un più profondo processing dei dati. I siti che si occupano di confrontare le liste della spesa sono spesso basati sullo scraping delle pagine dei fornitori in gioco e sul rappresentare i dati in modo tale da poter essere confrontati.

Lo svantaggio è che il processo di adattamento è un lavoro intenso, ed è difficile per l'utente finale acquisire le abilità necessarie. Una valutazione del lavoro è disponibile in Cunningham e Scott (2004); riferirsi pure all'articolo "Computational Language System, Architectures".

2.6.2 ACE: Automatic Content Extraction

Il programma Automatic Content Extraction (ACE, ACE (2004); Maynard et al. (2003)) è il successore di MUC che è stato lanciato da uno studio pilota nel 1999 e che ha continuato il ciclo di valutazione competitiva qualitativa del suo predecessore. ACE differisce da MUC per tre aspetti principali:

1) Molte delle cinque attività MUC definite nella sezione 4 sono combinate in ACE. NE e CO diventano un'attività unica in ACE denominata "Entity Detection and Tracking" (EDT), e TE e TR diventano in ACE l'attività "Relation Detection and Tracking". L'attività ST è rinominata "Event Detection and Characterisation".
2) Le attività ACE sono più complesse rispetto alle equivalenti MUC in molti aspetti. Nell'attività EDT, per esempio, esiste una tassonomia delle entità a grana più fine, ed è necessaria per i sistemi per interpretare riferimenti ad entità metonimiche, richiedendo un'analisi semantica del testo in considerazione. Inoltre, sono usati vari domini e sorgenti, inclusi gli output di trascrittori automatici di discorsi e programmi di identificazione ottica dei caratteri.
3) I risultati della valutazione del programma ACE non sono pubblici. Mentre in MUC tutti i risultati concorrenti venivano resi pubblici, i risultati ACE sono riservati ai partecipanti. L'utilità del programma ACE per i non partecipanti è dunque minore rispetto a quella MUC.

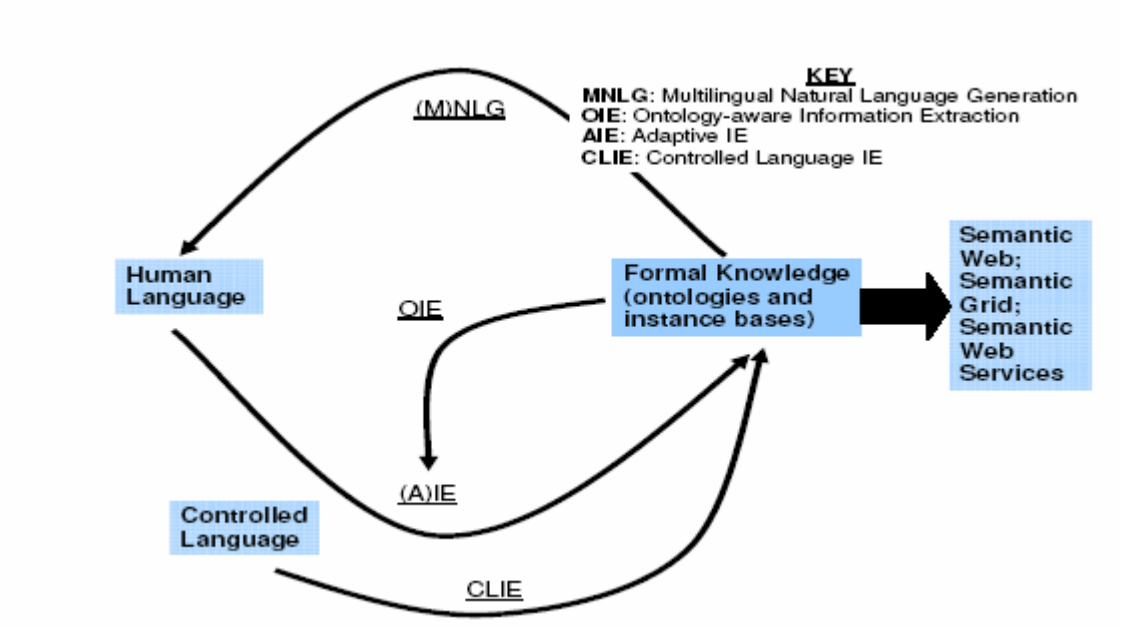


Figura 7. Closing the language loop

2.6.3 IE basata sulle Ontologie

Il Semantic Web ha intenzione di aggiungere uno strato maneggiabile e malleabile di annotazioni relative alle ontologie per integrare l'esistente web dell'ipertesto del linguaggio naturale Fensel et al.(2002); Davies et al.(2002); Bechhofer et al.(2003). Per permettere di realizzare questa visione, la creazione di annotazioni semantiche, il collegamento di pagine web alle ontologie, e la creazione, evoluzione ed interrelazione delle ontologie devono diventare processi automatici o semi-automatici, ed una parte significativa dei recenti lavori è stata indirizzata verso applicazioni di Ontology-Based IE (OBIE-Bontcheva (2004)) in questo contesto. La figura 7 illustra i modi nei quali IE ed altre tecnologie possono essere usati per mettere insieme il linguaggio naturale sul quale è principalmente basato il web di oggi e la conoscenza formale necessaria per il Sematic Web. OBIE pone due sfide principali:

- l'identificazione delle istanze di concetto dall'ontologia nel testo;
- il popolamento automatico delle ontologie con le istanze nel testo.

2.6.3.1 Identificazione delle istanze dalle ontologie

Se l'ontologia in questione è già stata popolata con delle istanze, l'attività di un sistema OBIE può essere semplicemente quella di identificare le istanze dall'ontologia nel testo. Metodologie simili possono essere usate per questo come per i tradizionali sistemi IE, usando un'ontologia piuttosto che un gazetteer piatto. Per sistemi basati su regole, questo è relativamente semplice. Per sistemi basati sull'apprendimento, comunque, questo è più problematico perché sono richiesti i training data. Collecting data e training data sono probabilmente dei colli di bottiglia. Diversamente dai tradizionali sistemi IE, per i quali i training data esistono in domini come testi di news in svariate forme, grazie agli sforzi di MUC, ACE e altri programmi, per le applicazioni di Semantic Web c'è scarsità di materiale disponibile. C'è bisogno che nuovi training data vengano creati manualmente o semi-automaticamente, ma ciò è un'attività onerosa e dispendiosa in termini di tempo; comunque sono in corso di sviluppo sistemi che vengano in soccorso, per esempio nella creazione di metadati.

2.6.3.2 Popolamento automatico delle ontologie

In questo caso un'applicazione OBIE identifica le istanze nel testo appartenenti ai concetti nell'ontologia, e aggiunge queste istanze all'ontologia nella corretta posizione. E' importante notare che le istanze possono comparire in più di una posizione nell'ontologia, a causa della natura multidimensionale di molte ontologie e/o all'ambiguità che non si può o non si vuole risolvere a questo livello.

Capitolo 3
GATE, a General Architecture for Text Engineering

In questo capitolo presentiamo il sistema di Information Extraction oggetto dell'analisi effettuata. Gate è un framework ed un ambiente grafico di sviluppo che permette agli utenti di sviluppare ed utilizzare componenti di language engineering in un modo robusto. L'architettura GATE ci ha permesso non solo di sviluppare un numero di applicazioni di successo per varie attività di language processing (come l'Information Extraction), ma anche di costruire ed annotare insiemi di documenti (corpora) e di effettuare valutazioni sull'applicazione generata. Il framework può essere usato per sviluppare applicazioni e risorse in svariate lingue, grazie al completo supporto Unicode.

3.1) Introduzione

Produrre componenti robusti per processare il linguaggio umano come parte di un'applicazione software richiede attenzione agli aspetti ingegneristici della loro costruzione. Questo capitolo riporta un lavoro realizzato con GATE[5], un'infrastruttura per lo sviluppo di software language processing che contribuisce su diversi fronti a questa predicibilità:

- il sistema è disegnato per separare chiaramente le attività di basso livello come il salvataggio o la visualizzazione dei dati, la localizzazione e il caricamento dei componenti e l'esecuzione dei processi dalle strutture di dati e dagli algoritmi che ad oggi analizzano il linguaggio umano;
- automatizzando le misurazioni del livello di performance dei componenti di language processing;
- riducendo le spese generali d'integrazione fornendo meccanismi standard per i componenti per comunicare i dati circa la lingua, e usando open standards come Java e XML come piattaforma sottostante;
- fornendo un set base di componenti di language processing che possono essere estesi o rimpiazzati dall'utente se richiesto.

Il resto del capitolo è strutturato come segue: nella sezione 3.2 verrà descritta l'architettura di GATE; nella sezione 3.3 verranno illustrati alcuni dettagli di applicazioni che sono state costruite usando GATE; la sezione 3.4 descrive le processing resources disponibili dentro GATE nel modulo ANNIE, mentre la sezione 3.5 descrive le language resources; nella sezione 3.6 si parlerà dei meccanismi per la valutazione e nella sezione 3.7 si discuterà delle direzioni future.

3.2) Un framework per applicazioni e strumenti robusti

GATE (Cunningham, 2002) è un'architettura, un framework ed un ambiente di sviluppo per LE (Language Engineering)[6]. Come *architettura*, definisce l'organizzazione di un sistema di LE e l'assegnazione delle responsabilità ai diversi componenti, ed assicura che

[5] Questo lavoro è stato supportato dall'Engineering and Physical Sciences Research Council (EPSRC) sotto le concessioni GR/K25267 e GR/M31699 e molte altre meno importanti.
[6] GATE è disponibile gratuitamente per il download dal sito http://gate.ac.uk.

l'interazione tra i componenti soddisfi i requisiti di sistema. Come *framework*, fornisce un design riutilizzabile per un sistema software di LE ed un set di blocchi per la creazione di software prefabbricati che il language engineer può usare, estendere e personalizzare per le sue specifiche necessità. Come *ambiente di sviluppo*, aiuta i suoi utenti a minimizzare il tempo speso a costruire nuovi sistemi di LE o a modificare quelli esistenti, aiutando lo sviluppo completo e fornendo un meccanismo di debug per i nuovi moduli. Poiché GATE possiede un modello basato sui componenti, permette un facile accoppiamento e disaccoppiamento dei processori, per mezzo di confronti facilitati di configurazioni alternative del sistema o di implementazioni differenti dello stesso modulo (es. diversi parser (analizzatori sintattici)). La disponibilità di strumenti per la comoda visualizzazione dei dati ad ogni punto del processo di sviluppo aiuta l'interpretazione immediata dei risultati.
Il framework GATE include una libreria centrale ed un set di librerie di LE riutilizzabili. Il framework implementa l'architettura e fornisce (tra le altre cose) facilitazioni per le risorse di processo e di visualizzazione, incluse la rappresentazione, l'import e l'export dei dati. I moduli riutilizzabili forniti con la libreria centrale sono capaci di svolgere le attività di processing della lingua di base come il POS tagging e il semantic tagging. Questo elimina la necessità dell'utente di ricreare lo stesso tipo di risorse, e fornisce un buon punto di partenza per nuove applicazioni. I moduli sono descritti in modo più dettagliato nelle sezione 3.4.
Le applicazioni sviluppate con GATE possono essere utilizzate al di fuori della Graphical User Interface (GUI), usando un accesso da linea di comando attraverso una GATE API (http://gate.ac.uk). In aggiunta, i moduli riutilizzabili e i componenti di visualizzazione possono essere tutti usati indipendentemente dall'ambiente di sviluppo. I componenti di GATE possono essere implementati da una varietà di linguaggi di programmazione e database, ma in ogni caso essi sono presentati al sistema come una classe Java. Questa classe può semplicemente chiamare il programma sottostante o fornire un livello di accesso a un database; in alternativa può implementare l'intero componente. Nel resto di questa sezione viene mostrato come le infrastrutture di GATE si prendono cura della scoperta, del caricamento e dell'esecuzione delle risorse, e brevemente si discute dell'immagazzinamento dei dati e della visualizzazione.

3.2.1 Algoritmi + dati + GUI = applicazioni

Il titolo esprime in modo succinto la distinzione fatta in GATE tra dati, algoritmi e il modo di visualizzarli. In altre parole, i componenti di GATE sono di tre tipi:

- **LanguageResources (LRs)** rappresentano le entità come lessici, corpora e ontologie;
- **ProcessingResources (PRs)** rappresentano le entità che sono fondamentalmente algoritmi, come i parser;
- **VisualResources (VRs)** rappresentano la visualizzazione e i componenti di editing che partecipano nelle GUI.

Queste risorse possono essere collocate nella macchina dell'utente o in remoto (disponibili via HTTP), e tutte possono essere estese dall'utente senza modificare GATE stesso.
Uno dei vantaggi principali del separare gli algoritmi dai dati di cui si servono è che i due possono essere sviluppati indipendentemente dai language engineers con differenti abilità, es. programmatori e linguisti. In modo del tutto simile, separare i dati dalla loro visualizzazione permette agli utenti di sviluppare visual resources alternative, usando

ancora una language resource fornita da GATE. In modo collettivo, tutte le risorse sono conosciute come CREOLE (a Collection of REusable Objects for Language Engineering), e sono dichiarate in un file XML "magazzino", che descrive il loro nome, implementando classe, parametri, icone, ecc. Questo file è usato dal framework per scoprire e caricare le risorse disponibili. Un tag dei parametri descrive i parametri di cui ogni risorsa ha bisogno quando viene creata o eseguita. I parametri possono essere opzionali, per esempio se una lista di documenti è fornita quando il corpus è costruito, esso verrà popolato automaticamente con questi documenti.
Quando un'applicazione viene sviluppata sotto l'ambiente grafico di GATE, l'utente sceglie quali processing resources inserire (es. tokeniser, POS tagger), in quale ordine dovranno essere eseguite e su quali dati (es. documento o corpus). Anche i parametri di esecuzione di ogni risorsa sono settati qui, per esempio un documento caricato viene passato come parametro ad ogni PR. Quando l'applicazione viene lanciata, i moduli verranno eseguiti sul documento passato nell'ordine specificato. I risultati si possono vedere nel viewer/editor dei documenti (figura 8).

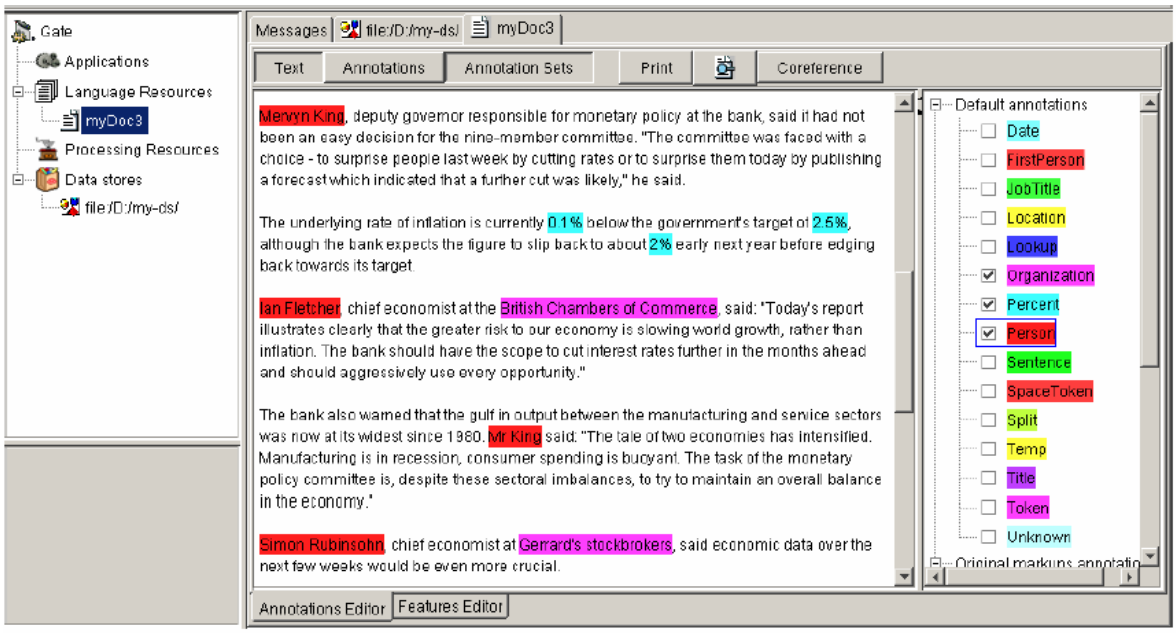


Figura 8. Viewer/editor di documenti di GATE

3.2.2 Rappresentazione e manipolazione dei dati

GATE supporta una varietà di formati inclusi XML, RTF, HTML, SGML, email e testi semplici. In tutti i casi, comunque, quando un documento viene creato o aperto in GATE, il formato è analizzato e convertito in un singolo modello unificato denominato *annotazione*. Questo formato è una forma modificata del formato TIPSTER (Grishman, 1997) che è stata resa largamente compatibile con il formato Atlas (Bird et al., 2000), ed usa l'attuale meccanismo standard dello "stand-off markup" (Thompson e McKelvie, 1997). Le annotazioni associate a ciascun documento sono una strutture centrale di GATE, poiché esse mettono in codice il linguaggio dei dati letti e producono per ognuno un modulo di processo. Il framework GATE fornisce anche un meccanismo di immagazzinamento persistente delle language resources. Attualmente offre tre tipi di immagazzinamento: uno usa i database relazionali (per esempio Oracle), mentre gli altri due sono basati sui file, usando la serializzazione Java o un formato interno basato su XML. I documenti di GATE

possono poi essere esportati nuovamente nel loro formato originale (es. SGML/XML per il British National Corpus (BNC)) e l'utente può scegliere se qualche altra annotazione aggiuntiva debba essere aggiunta ad esso oppure no. Per ricapitolare, l'esistenza di una struttura dati unificata assicura una comunicazione scorrevole tra i componenti, mentre il provvedere alle capacità di import ed export rende molto semplice la comunicazione con il mondo esterno.

3.2.3 Elaborazione multilingua

Negli ultimi anni l'enfasi attorno alla multilingualità è cresciuta, e progressi importanti sono stati rilevati nell'ambito software con l'emergere di Unicode come standard universale per la rappresentazione dei dati in forma testuale. GATE supporta l'elaborazione dei dati multilingua usando Unicode come default text encoder. Questo fornisce anche un significato dell'intero testo in svariate lingue, usando delle tastiere virtuali dove la lingua non è supportata dalla piattaforma operativa sottostante.(Da notare che sebbene Java rappresenti i caratteri come Unicode, esso non supporta l'input in molti dei linguaggi coperti da Unicode.) Attualmente sono supportate 28 lingue, e altre sono pianificate per le release future. Poiché GATE è un'architettura aperta, nuove tastiere virtuali possono essere definite dagli utenti ed aggiunte al sistema quando necessario. Per visualizzare il testo, GATE fa affidamento sui servizi di interpretazione offerti dall'implementazione Java per la piattaforma sulla quale gira.
L'abilità del manipolare i dati Unicode, assieme alla separazione tra dati e implementazione, permette ai sistemi LE basati su GATE di essere portati verso nuove lingue senza ulteriori costi, esclusi quelli per lo sviluppo delle risorse di cui si ha bisogno per la specifica lingua. Questi servizi sono stati sviluppati come parte del progetto EMILLE (McEnery et al., 2000), focalizzato sulla costruzione di un corpus elettronico comprendente 63 milioni di parole delle lingue dell'Asia meridionale.

3.3) Applicazioni

Uno dei punti forti di GATE è che esso è flessibile e robusto abbastanza da permettere lo sviluppo di un vasto range di applicazioni all'interno del suo framework. In questa sezione verranno descritte brevemente alcune delle applicazioni NLP sviluppate usando l'architettura GATE.

3.3.1 MUSE

Il sistema MUSE (Maynard et al., 2001) è un sistema per Named Entity recognition multi-purpose che è capace di elaborare testi appartenenti a domini e generi largamente diversi, aiutando in tal modo a ridurre la necessità di adattamenti costosi sia in termini di tempo che di denaro delle esistenti risorse alla nuove applicazioni e domini. Il sistema aiuta ad identificare i parametri rilevanti per la creazione di un sistema di name recognition attraverso i diversi tipi di mutevolezza, come i cambi nel dominio, nel genere e nel mezzo. Per esempio, i testi meno formali possono non seguire i formati standard delle maiuscole, della punteggiatura e dell'ortografia, e questo può essere un problema per molti generici sistemi NE. Le valutazioni attuali per questo sistema si aggirano intorno al 93% per la precision e al 95% per il recall frutto della media su una varietà di tipi di testi.

3.3.2 ACE[7]

Il sistema MUSE è stato anche adattato per prendere parte al corrente programma ACE (Automatic Content Extraction) a cura di NIST. Questo richiede sistemi per eseguire le attività di recognition e tracking di entità quali nomi, pronomi e le loro citazioni attraverso tre tipi di testi di notizie "puliti" (newswire, broadcast news e newspaper) e due tipi di testi di notizie "degradati" (output OCR e ASR).

3.3.3 MUMIS

Il sistema MUMIS (MUltiMedia Indexing and Searching environment) usa i componenti per l'Information Extraction sviluppati in GATE per produrre annotazioni formali circa gli avvenimenti essenziali nel materiale dei programmi video del football. Questo sistema IE include versioni del tokeniser, POS tagger, individuazione delle frasi e moduli del semantic tagging sviluppate come parte delle risorse standard di GATE, ma include anche moduli per l'analisi morfologica, per il completo parsine sintattico e per l'interpretazione del discorso, rendendo possibile in tal modo la produzione di annotazioni su testi incamerando informazioni strutturali, lessicali, sintattiche e semantiche. Attualmente il modulo di tagging semantico ottiene punteggi intorno al 95% per la precision e al 76% per il recall, che rappresentano un miglioramento significativo rispetto al sistema guida di named entity recognition utilizzato per la valutazione.

3.4) Processing Resources

GATE in principio fu sviluppato nel contesto della R&D dell'Information Extraction, e furono creati sistemi IE in molte lingue, forme e dimensioni usando GATE ed i componenti per l'IE distribuiti con esso[8].
GATE è distribuito con un sistema IE chiamato ANNIE, A Nearly-New IE system (sviluppato da Hamish Cunningham, Valentin Tablan, Diana Maynard, Kalina Bontcheva, Marin Dimitrov e altri).
ANNIE si basa su algoritmi a stati finiti e sul linguaggio JAPE. I suoi componenti formano una pipeline come viene mostrato in figura 9, e sono inclusi con GATE. Di seguito si illustrano questi componenti, caricati di default con ANNIE, più qualche altro modulo caricabile a parte.

3.4.1 Tokeniser

Il tokeniser fraziona il testo in simboli molto semplici come numeri, punteggiatura e parole di diversi tipi. Per esempio, viene effettuata la distinzione tra le parole con l'iniziale maiuscola (uppercase), quelle con l'iniziale minuscole (lowercase) e tra alcuni tipi di punteggiatura. Lo scopo è di limitare il lavoro del tokeniser verso la massima efficienza, e

[7] In questo paragrafo si richiama ed amplia ciò che è stato detto nel paragrafo 2.6.2

[8] I principali architetti dei sistemi IE nelle versione 1 di GATE furono Robert Gaizauskas e Kevin Humphreys. Questo lavoro vive nel sistema LaSIE. (Una derivazione di LaSIE fu distribuita con la versione 1 di GATE con il nome di VIE, a Vanilla IE system.)

raggiungere una maggiore flessibilità spostando il peso sulle regole grammaticali, che sono maggiormente adattabili.

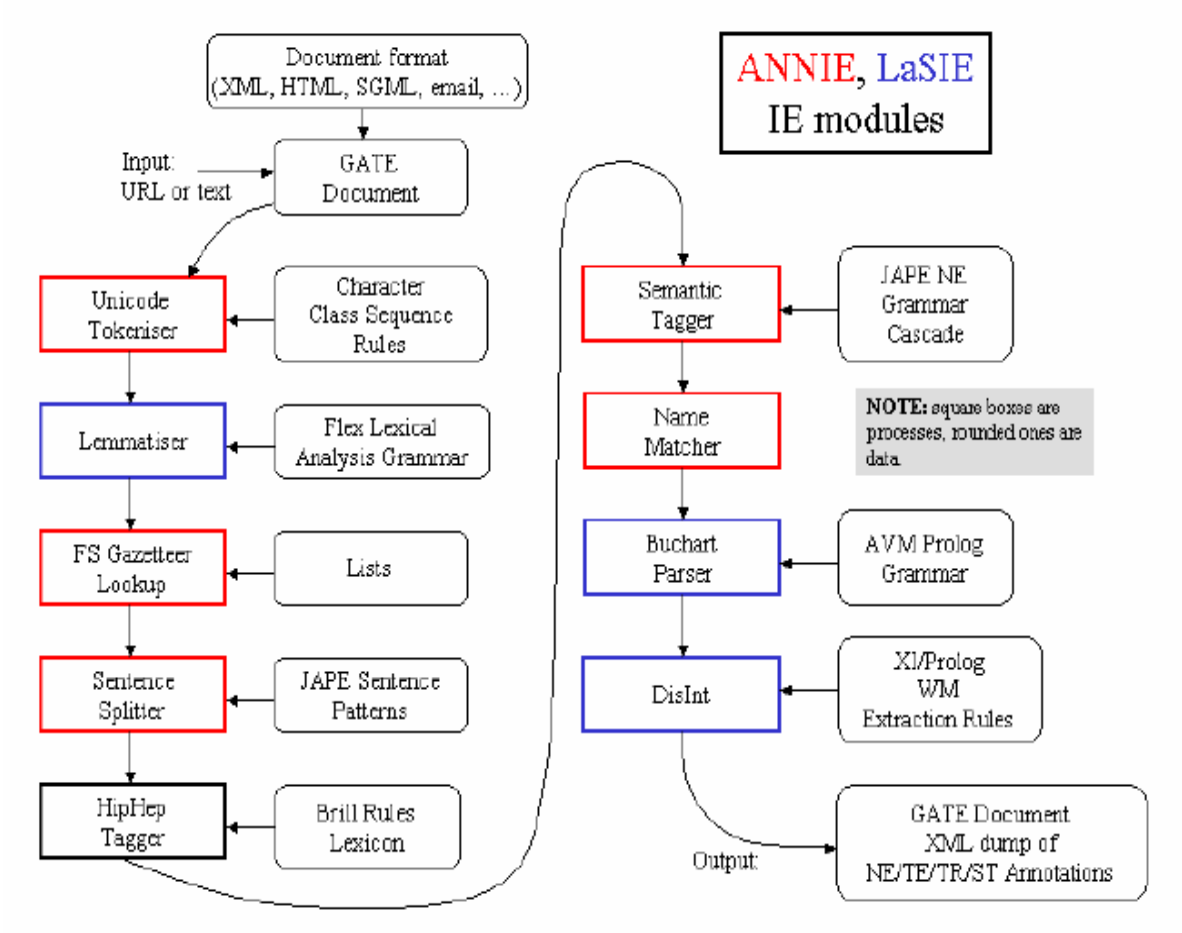


Figura 9. ANNIE e LaSIE

3.4.1.1 Regole del tokeniser

Una regola ha un left hand side (LHS) e un right hand side (RHS). Il LHS è un'espressione regolare che deve trovare riscontro con l'input; il RHS descrive le annotazioni che devono essere aggiunte all'AnnotationSet. Il LHS è separato dal RHS tramite il simbolo '>'. I seguenti operatori possono essere usati nel LHS:

|	(or)
*	(0 o più occorrenze)
?	(0 o 1 occorrenze)
+	(1 o più occorrenze)

Il RHS usa ';' come separatore, ed ha la seguente sintassi:

{LHS} > {Annotation type};{attribute1}={value1};...;{attribute n}={value n}

Ulteriori dettagli circa i costrutti primitivi disponibili sono presenti nel file del tokeniser (DefaultTokeniser.Rules).

La seguente regola è relativa ad una parola che inizia con una sola lettera maiuscola:

> *"UPPERCASE_LETTER" "LOWERCASE_LETTER"* > Token;orth=upperInitial; kind=word;*

Essa dichiara che la sequenza deve iniziare con una lettera maiuscola seguita da zero o più lettere minuscole. Questa sequenza verrà annotata come Token; l'attributo "orth" (orthography) ha il valore 'upperInitial' e l'attributo "kind" ha il valore 'word'.

3.4.1.2 Tipi di Token

Nel set di default delle regole, sono ammessi i seguenti tipi di Token e SpaceToken:

Word
Una word è definita come un qualsiasi set di lettere maiuscole e minuscole contigue, compreso il trattino d'unione (ma non qualsiasi altra forma di punteggiatura). Una word ha anche l'attributo "orth" per il quale sono previsti quattro tipi di valori:

- upperInitial – la lettera iniziale è maiuscola, il resto della parola è minuscolo
- allCaps – tutte le lettere sono maiuscole
- lowerCase – tutte le lettere sono minuscole
- mixedCaps – ogni composizione di lettere maiuscole e minuscole non compresa nei valori precedenti

Number
Un number è definito come una qualsiasi combinazione di cifre consecutive. Non sono previste sottocategorie per il number.

Symbol
Sono stati definiti due tipi di symbol: symbol di valute (es. '£', '$', '€') e symbol (es. '&', '^'). Sono rappresentati da qualsiasi numero consecutivo di entrambe le categorie.

Punctuation
Sono stati definiti tre tipi di punctuation: start_punctuation (es. '('), end_punctuation (es. ')') e other_punctuation (es. ':'). Ogni elemento di punctuation è un token separato.

SpaceToken
Gli spazi bianchi vengono divisi in due tipi di SpaceToken – space e control – a seconda che siano puri caratteri di spazio o caratteri di controllo. Ogni set continuo (ed omogeneo) di caratteri di spazio o di controllo è definito come SpaceToken.

La descrizione effettuata sopra si riferisce al tokeniser di default. Comunque, tokeniser alternativi possono essere creati se necessario. La scelta del tokeniser viene poi effettuata al momento dell'elaborazione del testo.

3.4.1.3 English Tokeniser

L'English Tokeniser è una processing resource che comprende un normale tokeniser e un traduttore JAPE. Il traduttore ha il compito di adattare l'output generico del tokeniser ai requisiti del part-of.speech tagger English. Un tipico adattamento può essere il mettere insieme in un unico token costrutti del tipo " '30s", " 'Cause", " 'em", " 'll", " 're", " 'til", " 've". Un'altra attività del traduttore JAPE è di convertire i costrutti negativi come "don't" da tre token ("don", " ' " e "t") a due token ("do" e "n't").
L'English Tokeniser dovrebbe sempre essere usato sui testi inglesi che hanno bisogno di essere elaborati in seguito dal POS Tagger.

3.4.2 Gazetteer

Le liste gazetteer usate in GATE sono semplici file di testo, con una entry per riga. Ogni lista rappresenta un set di nomi, come per esempio nomi di città, di organizzazioni, giorni della settimana, ecc. Queste liste non consistono solo di entità, ma anche di nomi di *indicatori* pratici, come tipiche designazioni di compagnie (es. Ltd), titoli, ecc..
Di seguito viene riportata una piccola sezione della lista relativa alle monete:

Ecu
European Currency Units
FFr
Fr
German mark
German marks
New Taiwan dollar
New Taiwan dollars
NT dollar
NT dollars

Per accedere a queste liste si usa un index file (lists.def); per ogni lista, è specificato un tipo principale e, opzionalmente, un tipo minore[9]. Nell'esempio che segue, la prima colonna si riferisce al nome della lista, la seconda colonna al tipo principale, e la terza al tipo minore. Queste liste sono compilate dentro automi a stati finiti. Ogni token del testo che trova riscontro in questi automi a stati finiti verrà annotato con caratteristiche specificanti i tipi principale e minore. Le regole grammaticali poi specificano come identificare i tipi in particolari circostanze. Ogni lista gazetteer dovrebbe risiedere nella stessa directory dell'index file.

currency_prefix.lst:currency_unit:pre_amount
currency_prefix.lst:currency_unit:post_amount
date.lst:date:specific
day.lst:date:day

Così, se per esempio bisogna identificare uno specifico giorno, bisogna specificare nella grammatica il tipo minore "day", in modo da confrontare solo le informazioni relative ai giorni specifici; se invece si vuole identificare ogni tipo di data, bisogna specificare il tipo

[9] E' anche possibile includere una lingua nello stesso modo, dove vengono usate liste differenti per le lingue differenti, sebbene ANNIE sia stato concepito con la recognition monolingua.

principale “date”, per permettere ai token annotati con ogni informazione relativa ad una data di essere identificati.

3.4.3 Sentence Splitter

Il sentence splitter è una cascata di traduttori a stati finiti che divide il testo in periodi. Questo modulo è un requisito per il tagger. Lo splitter usa una lista gazetteer di abbreviazioni per aiutarsi a distinguere i punti di fine frase dagli altri tipi.
Ogni periodo viene annotato con il tipo Sentence. Ad ogni simbolo di interruzione del periodo (per esempio un punto) viene data anche un’annotazione “Split”. Questa ha molti possibili tipi: “.”, “punctuation”, “CR” (carattere terminatore di linea) o “multi” (una serie di simboli di punteggiatura come “!?!”). Il sentence splitter è indipendente dal dominio e dall’applicazione.

3.4.4 Part of Speech Tagger

Il tagger [Hepple 00] è una versione modificata del tagger Brill, che produce un part-of-speech tag come un’annotazione su ogni parola o simbolo. Il tagger usa un dizionario di default e un set di regole, frutto dell’esercitazioni su un corpus preso dal Wall Street Journal. Entrambi possono essere modificati manualmente se necessario. Esistono due dizionari aggiuntivi – uno per i testi scritti tutti in maiuscolo (lexicon_cap) e uno per i testi scritti tutti in minuscolo (lexicon_lower). Per poterli usare, il dizionario di default deve essere rimpiazzato con quello appropriato nel momento del caricamento. Il set di regole invece è unico e quindi viene utilizzato sempre quello.
Il Part-of-Speech tagger di ANNIE richiede i seguenti parametri:

- encoding – codifica usata per le regole di lettura e per i dizionari (init-time)
- lexiconURL – l’URL per il file col dizionario (init-time)
- rulesURL – l’URL per il file con il set di regole (init-time)
- document – il documento che viene processato (run-time)
- inputASname – il nome del set di annotazioni usato come input (run-time)
- outputASname – il nome del set di annotazioni usato come output (run-time). Questo è un parametro opzionale. Se l’utente non fornisce nessun valore, nuove annotazioni vengono create sotto il set di default
- baseTokenAnnotationType – il nome del tipo di annotazione che si riferisce ai Token in un documento (run-time, default = Token)
- baseSentenceAnnotationType – il nome del tipo di annotazione che si riferisce alle Sentences in un documento (run-time, default = Sentences)
- outputAnnotationType – tag POS che sono aggiunti come caratteristiche della categoria sulle annotazioni del tipo “outputAnnotationType” (run-time, default = Token)

If – (inputASname == outputASname) AND (outputAnnotationType == baseTokenAnnotationType)

then – nuove caratteristiche sono aggiunte alla annotazioni esistenti del tipo “baseTokenAnnotationType”.

otherwise – il tagger cerca le annotazioni del tipo "outputAnnotationType" sotto il set di annotazioni "outputASname" che ha gli stessi offset dell'annotazione del tipo "baseTokenAnnotationType". Se ciò accade, aggiunge nuove caratteristiche all'annotazione trovata o, altrimenti, crea una nuova annotazione del tipo "outputAnnotationType" sotto il set di annotazioni "outputASname".

3.4.5 Semantic Tagger

Il Semantic Tagger consiste di regole scritte nel linguaggio JAPE (Java Annotations Pattern Engine) (Cunningham et al., 2002), che descrive le modalità per trovare i riscontri e i modi in cui creare le annotazioni come risultato. JAPE è una versione di CPSL (Common Pattern Specification Language) (Applet, 1996) che fornisce traduzioni a stati finiti su annotazioni basate su espressioni regolari. Una grammatica JAPE consiste di un set di fasi, ognuna delle quali composta da un set di modelli/action rules, che vengono lanciati sequenzialmente. I modelli possono essere specificati descrivendo una specifica stringa di testo, o annotazioni precedentemente create da moduli come il tokeniser, gazetteer, o analisi del formato di un documento. La prioritizzazione (se attivata) previene l'assegnamento multiplo delle annotazioni alla stessa stringa di testo.

3.4.6 Orthographic Coreference (OrthoMatcher)[10]

Il modulo Orthomatcher aggiunge relazioni d'identità tra le named entities trovate dal semantic tagger, in modo tale da poter realizzare la coreference (riferimento). Esso non trova nuove named entities, ma può assegnare un tipo ad un nome proprio non classificato, utilizzando il tipo del nome che realizza il match.
Le regole per il riscontro sono invocate solo se i nomi che vengono comparati sono dello stesso tipo, per esempio entrambi già etichettati come organizzazioni, o se uno dei due è classificato come "unknown". Questo evita che i nomi precedentemente classificati vengano ricatalogati.

3.4.7 Pronominal Coreference

Questo modulo realizza la risoluzione anaforica utilizzando i formalismi grammaticali di JAPE. Da notare che questo modulo non viene automaticamente caricato con gli altri moduli di ANNIE, ma può essere caricato separatamente come Processing Resource. Il modulo principale consiste di tre submoduli:

- quoted text module
- pleonastic it module
- pronominal resolution module

I primi due moduli sono submoduli di aiuto per quello pronominale, poiché essi non realizzano niente relativo alla coreference resolution eccetto la localizzazione di frammenti quotati e occorrenze pleonastiche di it nel testo. Essi generano annotazioni temporanee che sono utilizzate dal submodulo pronominale (tali annotazioni temporanee vengono rimosse in seguito).

[10] Una ulteriore denominazione è Name Matcher.

Il modulo coreference principale può operare con successo solo se tutti i moduli di ANNIE sono stati già eseguiti. Il modulo dipende dalle seguenti annotazioni create dai rispettivi moduli di ANNIE:

- Token (English Tokenizer)
- Sentence (Sentence Splitter)
- Split (Sentence Splitter)
- Location (NE Trasducer, OrthoMatcher)
- Person (NE Trasducer, OrthoMatcher)
- Organization (NE Trasducer, OrthoMatcher)

Per ogni pronome (anafora) il modulo coreference genera un'annotazione del tipo "Coreference" contenente due caratteristiche:

- offset antecedente – è l'offset del nodo iniziale per l'annotazione (entità) che è proposta come antecedente, o è null se nessuna antecedente viene proposta.

- matches – è una lista di IDs relativi alle annotazioni che include la catena di coreference compresa la coppia anafora/antecedente.

3.4.7.1 Quoted Speech Submodule

Questo modulo identifica frammenti quotati nel testo che si sta analizzando. I frammenti identificati vengono utilizzati dal modulo di coreference pronominale per la giusta risoluzione di pronomi quali I, me, my, ecc. che compaiono nei frammenti quotati del testo. Il modulo produce annotazioni "Quoted Text". Il submodulo stesso è un traduttore JAPE che carica una grammatica JAPE e costruisce una FSM su di essa. La FSM ha il compito di trovare i frammenti quotati e di generare annotazioni appropriate che saranno utilizzate dopo dal modulo pronominale.
La grammatica JAPE consiste di sole quattro regole, che creano annotazioni temporanee per tutta la punteggiatura segnalata che può includere discorsi quotati, come ", ',". Queste regole provano dopo ad identificare i frammenti racchiusi da tale punteggiatura. Alla fine tutte le annotazioni temporanee generate durante il processo, eccetto quelle del tipo "Quoted Text" vengono rimosse (poiché nessun altro modulo se ne servirà più).

3.4.7.2 Pleonastic It Submodule

Il modulo va alla ricerca delle occorrenze pleonastiche del pronome "it". In modo del tutto simile al quoted speech submodule, è un traduttore JAPE che opera con una grammatica contenente modelli che trovano riscontro nei più comuni costrutti pleonastici relativi ad "it" osservati.

3.4.7.3 Pronominal Resolution Submodule

La principale funzionalità del modulo di coreference resolution è visibile nel submodulo di risoluzione pronominale. Esso si serve dei risultati derivanti dall'esecuzione dei submoduli quoted speech e pleonastic it. Il modulo lavora in accordo con il seguente algoritmo:

- Preprocessa il documento corrente. Questo step localizza le annotazioni di cui il submodulo ha bisogno (come Sentence, Person, Token, ecc.) e prepara la appropriate strutture dati per queste;

- Per ogni pronome esegue le seguenti azioni:

  - esamina il contesto appropriato relativo a tutti gli antecedenti candidati per ogni tipo di pronome;

  - sceglie l'antecedente migliore (se ne è presente qualcuno).

- Crea la catena di coreference dalle coppie individuali anafora/antecedente e le informazioni di coreference fornite dall'OrthoMatcher (questo step è eseguito dal modulo principale di coreference).

3.4.7.4 Descrizione dettagliata dell'algoritmo

Viene di seguito presentato in modo dettagliato l'algoritmo relativo alla pronominal coreference.

Preprocessing
L'attività di preprocessing include le seguenti sotto-attività:

- Identificazione delle frasi nel documento che viene processato. Le frasi sono identificate con l'aiuto delle annotazioni Sentence generate dal Sentence Splitter. Per ogni frase viene preparata una struttura dati contenente tre liste. Le liste contengono le annotazioni per le named entities person/organization/location che appaiono nella frase. Queste sono identificate grazie all'aiuto delle annotazioni Person, Organization e Location già generate dal Named Entity Trasducer e dall'OrthoMatcher.

- Il genere di ogni persona presente nella frase è identificato ed immagazzinato in una struttura dati globale. E' possibile che l'informazione relativa al genere non sia presente per alcune entità – per esempio se è presente solo il nome di famiglia (cognome) il Named Entity trasducer non sarà capace di dedurre il genere. In questi casi la lista con le entità derivate dal confronto generata dall'OrthoMatcher viene ispezionata e se qualcuno dei match ortografici contiene informazioni sul genere, queste vengono assegnate all'entità che si sta processando.

- Le occorrenze identificate di it pleonastico vengono salvate in una lista separata. Le annotazioni "Pleonastic it" generate dal submodulo pleonastico sono utilizzate per l'attività.
- Per ogni frammento di testo quotato, identificato dal quoted text submodule, viene creata una speciale struttura che contiene la persona ed i pronomi di terza persona singolare come "he" e "she" che compaiono nella frase contenente il testo quotato, ma non nell'intervallo d testo quotato (es. quelli che precedono e susseguono il testo quotato).

Risoluzione dei pronomi
Questa attività implica le seguenti sotto-attività:

ricerca dei pronomi nel documento. I pronomi sono rappresentati come annotazioni del tipo "Token" con la caratteristica "category" avente valore "PRP" o "PRP$". Il primo valore classifica gli aggettivi possessivi come my, your, ecc., il secondo classifica i pronomi personali e riflessivi. I due tipi di pronomi vengono combinati in una lista ed ordinati in accordo al loro offset nel testo.
Per ogni pronome presente nella lista vengono eseguite le seguenti azioni:

- Se il pronome è "it", viene eseguito un controllo sulla presenza di un'occorrenza pleonastica e, se così, non viene fatto nessuno sforzo ulteriore per la risoluzione.

- Viene determinato il contesto proprio. La dimensione del contesto viene determinata dal numero di frasi che conterrà. Il contesto include sempre la frase corrente (quella contenente il pronome), la frase precedente e zero o più frasi successive.
- A seconda del tipo di pronome, viene proposto un set di antecedenti candidati. Questo set include le named entities che sono compatibili con quel pronome. Per esempio, se il pronome in questione è "she", solo le annotazioni di tipo "Person" con la caratteristica "gender" uguale a "female" o "unknow" verranno considerate come candidate.

- Tra tutti i candidati, solo uno viene scelto in accordo con i criteri di valutazione specifici del pronome.

Generazione della catena di coreference
Questo step è effettivamente eseguito dal modulo principale. Dopo aver eseguito ognuno dei submoduli sul documento corrente, il modulo di coreference segue i seguenti passi:

- Trova le coppie anafora/antecedente generate dai submoduli.

- Per ogni coppia, viene trovato il riscontro ortografico (se presente) dell'entità antecedente e poi esteso con l'anafora della coppia (es. il pronome). Il risultato è la catena di coreference per l'entità. La coreference chain contiene gli IDs delle annotazioni (entità) sulle quali sta avvenendo la coreference.

- Una nuova annotazione Coreference viene creata per ogni catena. Questa annotazione contiene una singola chiave "matches" il cui valore è la catena di coreference (la lista con gli IDs). Le annotazioni vengono esportate in un annotation set pre-specificato.

La risoluzione di she, her, her$, he, him, his, himself e herself è simile poiché l'analisi del corpus ha evidenziato che questi pronomi sono correlati ai loro antecedenti in modo simile. Le caratteristiche del processo di risoluzione sono:

- Il contesto ispezionato non è molto grande – i casi in cui l'antecedente viene trovato oltre la terza frase precedente rispetto all'anafora sono rari.

- Il fattore recency è pesantemente usato – agli antecedenti candidati che nel testo appaiono più vicino all'anafora viene assegnato un punteggio maggiore.

- La anafore hanno una priorità più alta rispetto alle catafore. Se c'è un candidato anaforico ed uno cataforico, viene preferito quello anaforico, anche se quello cataforico ha un punteggio più alto.

Il processo di risoluzione si può riassumere nei seguenti passi:

- Ispezionare il contesto dell'anafora per trovare gli antecedenti candidati. Come candidato è considerata ogni annotazione del tipo Person. I casi in cui she/her si riferiscono ad entità inanimate (per esempio ship) non vengono considerati.

- Per ogni candidato viene eseguito un controllo di compatibilità del genere - solo i candidati aventi la caratteristica "gender" uguale ad "unknow" o compatibile con il pronome vengono considerati nell'analisi successiva.

- Valutare ogni candidato con il miglior candidato lontano. Se i due candidati sono anaforici per il pronome, allora scegliere quello che compare più vicino. Lo stesso avviene nei casi in cui i due candidati sono cataforici rispetto al pronome. Se un candidato è anaforico e l'altro cataforico, scegliere il primo, anche se quello cataforico compare più vicino al pronome.

Risoluzione di it, its, itself
Questo set di pronomi condivide molte caratteristiche comuni. Il processo di risoluzione contiene alcune differenze rispetto a quello visto per i precedenti insiemi di pronomi. Una risoluzione di successo di it, its, itself è resa più difficile dai seguenti fattori:

- Non ci sono restrizioni sulla compatibilità del genere. Nel caso in cui ci siano molti candidati nel contesto, la restrizione sulla compatibilità del genere diviene molto utile per eliminare qualche candidato. Quando non esiste nessuna restrizione, e con la mancanza di qualsiasi informazione sintattica od ontologica, il ruolo principale per la scelta del migliore antecedente viene giocato dal fattore recency.

- Il numero degli antecedenti nominali (es. entità a cui si riferisce non tramite un nome) è di molto più alto rispetto a quello relativo ai pronomi she, he, ecc. In questo caso cercare di trovare l'antecedente solo tra le named entities degrada di molto la precisione.

Risoluzione di I, me, my, myself
La risoluzione di questi pronomi dipende dal lavoro svolto dal submodulo quoted speech. Una importante differenza rispetto al processo di risoluzione degli altri pronomi è che il contesto non viene misurato in frasi ma dipende unicamente dall'ampiezza del periodo quotato. Un'altra differenza è che il periodo non è contiguo – lo stesso frammento quotato è escluso dal contesto, poiché è improbabile che un antecedente per I, me, my, myself appaia lì. Il contesto stesso è composto da:

- la parte della frase dove il frammento quotato origina, che non è contenuto nelle virgolette – es. il testo prima delle virgolette;

- la parte della frase dove il frammento quotato termina, che non è contenuto nelle virgolette – es. il testo che segue le virgolette;

- la parte della frase precedente il periodo nel quale le virgolette iniziano, che non è incluso in altre virgolette.

Vale la pena notare che contrariamente agli altri pronomi, l'antecedente di I, me, my, myself è più spesso cataforico, o se anaforico non è nello stesso periodo del frammento quotato.

L'algoritmo di risoluzione consiste nei seguenti step:

- Localizzare la descrizione del frammento quotato che contiene il pronome. Se il pronome non è contenuto in nessun frammento allora viene restituito il controllo senza proporre un antecedente.

- Ispezionare il contesto del frammento quotato (come definito prima) alla ricerca degli antecedenti candidati. I candidati sono considerati annotazioni del tipo Person o annotazioni del tipo Token con la caratteristica category = "PRP", string = "she" oppure category = "PRP", string = "he".

- Provare a cercare un candidato nel testo seguente il frammento quotato (primo modello). Se è presente più di un candidato, scegliere quello più vicino alla fine delle virgolette. Una volta trovato il candidato, proporlo come antecedente ed uscire.

- Provare a cercare un candidato nel testo precedente il frammento quotato (terzo modello). Scegliere il più vicino all'inizio delle virgolette. Una volta trovato il candidato, proporlo come antecedente ed uscire.

- Provare a cercare gli antecedenti nella parte non quotata della frase precedente il periodo dove iniziano le virgolette (secondo modello). Dare la preferenza a quello più vicino alla fine delle virgolette (se presenti) della frase precedente o a quello più vicino all'inizio del periodo.

3.4.8 Implementazione

L'implementazione delle processing resources è incentrata sulla robustezza, utilizzabilità e la chiara distinzione tra la rappresentazione dichiarativa dei dati e gli algoritmi a stati finiti. Il comportamento di tutti i processi è completamente controllato da risorse esterne come le grammatiche o i set di regole, che rendono i processi facilmente modificabili dagli utenti senza il bisogno di essere familiari con i linguaggi di programmazione. Il fatto che tutte le processing resources usino la tecnologia dei traduttori a stati finiti le rende piuttosto performanti in termini di tempo d'esecuzione. Esperimenti iniziali mostrano che il sistema completo di named entity recognition è in gradi di processare intorno a 2.5 KB/s su un PIII 450 con 256 MB di RAM (indipendentemente dalla dimensione del file di input; il requisito di processing è lineare rispetto alla dimensione del file di testo). La scalabilità è stata testata facendo girare i moduli di ANNIE su una parte scelta a caso del British National Corpus (circa il 10% dei documenti), che conteneva più di 17 MB di documenti.

3.5) Creazione delle language resources

Poiché molti algoritmi NLP richedono corpora annotati per l'"allenamento", l'ambiente di sviluppo di GATE fornisce facilitazioni semplici da usare ed estendibili per il text annotation. Per permettere di testare la loro usabilità nella pratica, furono usate queste facilitazioni per costruire corpora di testi *named entity annotated* per le applicazioni MUSE, ACE e MUMIS. L'annotazione può venir creata manualmente dall'utente o semi-automaticamente lanciando alcune processing resources sul corpus e poi andando a correggere/aggiungere manualmente. A seconda delle informazioni che bisogna annotare, qualche modulo di ANNIE può essere usato o adattato per "arrangiare" l'attività di annotazione sul corpus. Per esempio, utenti provenienti dalle discipline classiche crearono una gazetteer list contenente nomi di posti relativi alla Londra del 18° secolo, che venne fornita al gazetteer di ANNIE, consentendo l'annotazione automatica delle informazioni relative ai luoghi presenti in una vasta collezione di verbali di corte del 18° secolo provenienti dal Old Bailey di Londra.
Poiché l'annotazione manuale è un'attività difficile e propensa all'errore, GATE cerca di renderla semplice da effettuare mantenendo il suo carattere flessibile. Per aggiungere una nuova annotazione, l'utente seleziona con il mouse il testo interessato (es. Mr. "Clever") e poi clicca sul tipo di annotazione d'interesse (es. "Person"), che viene mostrata nella parte destra della schermata di visualizzazione del documento. Se comunque il tipo di annotazione desiderato non appare in quella lista o l'utente vuole associare informazioni più dettagliate all'annotazione (non solo il tipo), si può usare un *annotation editing dialogue*.

3.6) Salvataggio dell'output

Ci sono tre modi principali di salvare su file i risultati di un processo di Information Extraction eseguito su un documento:

1. salvare la Language Resource in un Datastore;

2. salvare l'output nel formato di serializzazione di GATE XML (includendo tutte le annotazioni del documento);

3. preservare il formato originale del documento, con l'aggiunta opzionale delle annotazioni.

Per salvare un testo in un Datastore, bisogna che un nuovo Datastore venga creato se non esiste già. Per fare ciò, basta cliccare con il tasto destro sulla voce "Datastore" e selezionare l'opzione "Crea un nuovo Datastore" selezionando il tipo di Datastore che si vuole creare e creando una directory da usare come Datastore (poiché un Datastore non è un file ma una directory). A questo punto, per salvare un documento basta cliccare su di esso e selezionare "Save to…".
Per aprire una Language Resource salvata in un Datastore, non bisogna cercare di caricarla come LR ma bisogna selezionare l'opzione "Open Datastore".

Selezionando invece la voce "Save as XML.." il nostro documento viene esportato in un file XML. Ricaricando tale documento, lo stato corrente viene ristabilito. Da notare che a

causa della ricchezza maggiore del modello di annotazione di GATE rispetto all'XML, lo stato non sarà del tutto identico dopo il ripristino. Se l'intento è quello di salvare il documento per un utilizzo futuro, è più indicato utilizzare il Datastore.

La terza opzione di salvataggio invece permette di preservare il formato originale del documento che si sta salvando. A tale file possono essere poi aggiunte delle annotazioni: se un documento è aperto nel viewer di GATE e qualche categoria delle entità è selezionata, scegliendo questa modalità il file viene salvato solo con quel tipo di annotazioni. Questo è il modo migliore di usare il sistema per aggiungere informazioni derivanti da qualche processo di analisi: si ha un file identico all'originale con l'aggiunta delle sole annotazioni selezionate.

Da non trascurare il fatto che le annotazioni in GATE sono strutturate secondo un grafo, che è difficile da rappresentare in XML (poiché è un formato di rappresentazione strutturato ad albero). Perciò, le annotazioni che nel salvataggio incontrano questa difficoltà vengono scartate generando un messaggio di avviso.

3.7) Valutazione delle prestazioni

La parte vitale di ogni applicazione relativa alla *language engineering* è la valutazione delle sue performance, ed un ambiente di sviluppo per questo scopo non sarà mai completo senza un meccanismo per la sua misurazione in un vasto insieme di casi di test. GATE contiene due meccanismi del genere: uno strumento di valutazione (AnnotationDiff) che consente misurazioni automatiche delle performance e la visualizzazione dei risultati, ed uno strumento di benchmarking, che permette il tracciamento dei progressi e dei regressi dei test sul sistema.

3.7.1 AnnotationDiff Tool

Lo strumento AnnotationDiff di GATE consente di confrontare due set di annotazioni create sullo stesso documento, per permettere o di paragonare un testo annotato automaticamente da un sistema con un testo referenziato (annotato manualmente), o di confrontare l'output di due versioni differenti dello stesso sistema (o di due sistemi diversi).
Per ogni tipo di annotazione, un numero rappresentativo viene generato per precision, recall, F-measure e false-positive[11]. Ognuno di questi può essere calcolato rispettando 3 criteri diversi – strict (esatto), lenient (clemente) ed average (media). La differenza sta nel considerare le risposte parzialmente corrette in modi diversi:

- La misura strict considera tutte le risposte parzialmente corrette come incorrette (spurie).
- La misura lenient considera tutte le risposte parzialmente corrette come corrette.
- La misura average realizza una via di mezzo relativamente alle risposte parzialmente corrette (per esempio effettua la media tra strict e lenient).

[11] La definizione di questi termini verrà fornita nel paragrafo 3.6.4.

La parte grafica dell'AnnotationDiff mostra i due set di annotazioni, evidenziati con due colori differenti. Le annotazioni appartenenti al key set hanno due possibili colori a seconda del loro stato[12]: bianco per le annotazioni che presentano un'annotazione compatibile (o parzialmente compatibile) nel response set, e arancione per le annotazioni che sono mancanti nel response set. Le annotazioni del response set hanno tre possibili colori: verde se sono compatibili con l'annotazione key, blu se sono parzialmente compatibili, e rosso se sono spurie. Nel visualizzatore, due annotazioni vengono posizionate sulla stessa riga se sono co-estensive, oppure su righe diverse se non lo sono.
Quando viene eseguito il confronto, prima vengono presi in considerazione l'offset e le features delle annotazioni e , dopo ciò, inizia il processo di confronto vero e proprio. Tutte le annotazioni del key set vengono paragonate con quelle del response set, e quelle che si è trovato avere lo stesso offset di inizio e fine vengono visualizzate sulla stessa linea nella tabella. Dopo, l'AnnotationDiff valuta se le features di ogni annotazione del response set comprendono quelle del key set, come specificato dal parametro keyFeatureNamesSet.

3.7.2 Le sei relazioni tra annotazioni

Coestensiva
Due annotazioni sono coestensive se relativa allo stesso frammento di testo in un documento. Fondamentalmente, i loro offset di inizio e di fine coincidono.
Sovrapposta
Due annotazioni sono sovrapposte se coprono un frammento di testo in comune

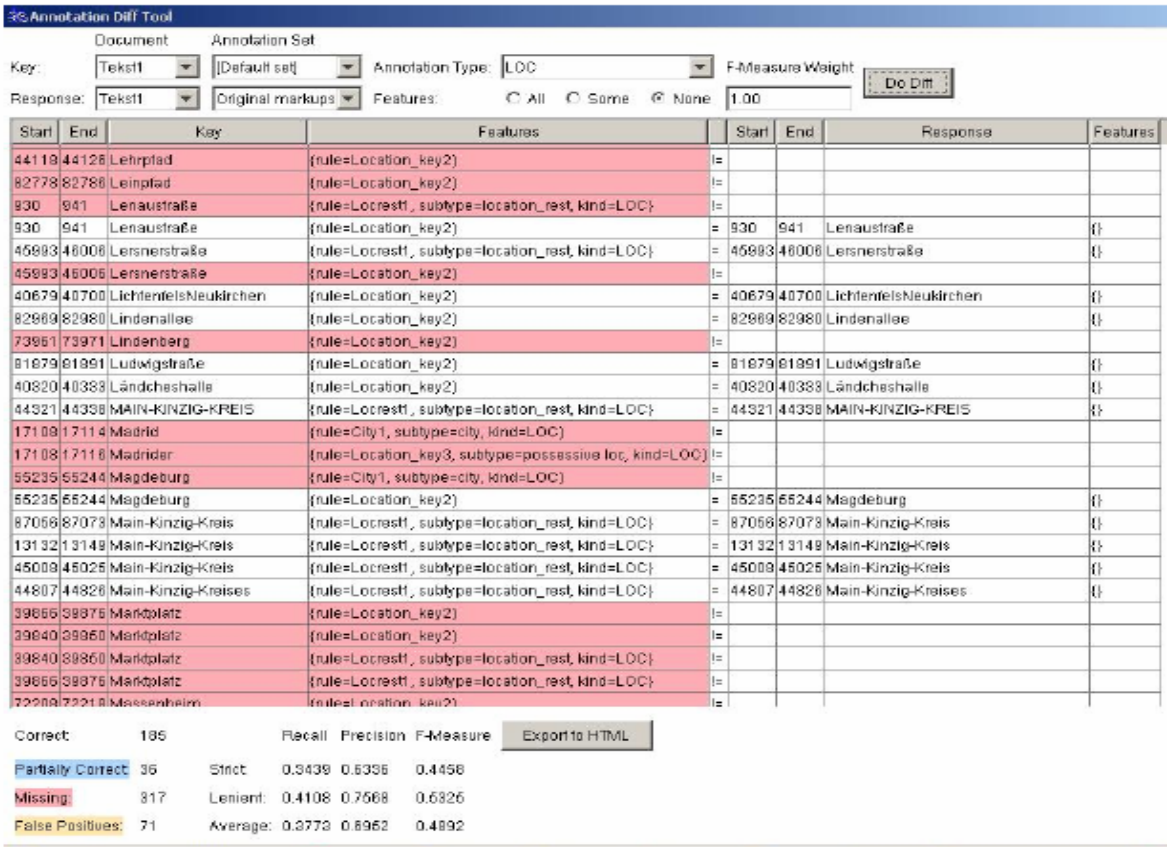


Figura 10. AnnotationDiff viewer

[12]Gli stati utilizzati in questo paragrafo verranno illustrati in quello successivo.

Compatibile (Corretta)
Due annotazioni sono compatibili se sono coestensive e se le features di una (di solito quella proveniente dal key set) sono contenute nelle features dell'altra (solitamente quella proveniente dal response set).
Parzialmente Compatibile (Parzialmente Corretta)
Due annotazioni sono parzialmente compatibili se sono sovrapposte e se le features di una (di solito quella proveniente dal key set) sono contenute nelle features dell'altra (solitamente quella proveniente dal response set).
Mancante
Si applica solo alle annotazioni key. Un'annotazione key viene definita mancante o se non è né coestensiva né sovrapposta, o se una o più features non sono contenute nell'annotazione response.
Spuria
Si applica solo alle annotazioni response. Un'annotazione response viene definita spuria o se non è né coestensiva né sovrapposta, o se una o più features dell'annotazione key non sono contenute nell'annotazione response.

3.7.3 Benchmarking Tool

Lo strumento di Benchmarking differisce dall'AnnotationDiff nel fatto che permette di effettuare la valutazione su un intero corpus invece che su di un singolo documento. Permette anche il tracciamento delle performance del sistema nel tempo.
Lo strumento richiede una versione pulita del corpus (cioè senza annotazioni) ed un corpus annotato a mano. Prima di tutto, lo strumento viene eseguito in generation mode per far si che crei un set di testi annotati dal sistema. Questi testi vengono immagazzinati per usi futuri. Il tool può essere poi lanciato in tre modi diversi:

1) confrontando il set processato salvato con il set annotato manualmente;
2) confrontando il corrente set processato con il set annotato manualmente;
3) (default mode) confrontando il set processato salvato con il corrente set processato e il set annotato manualmente.

In ogni caso, verranno riportate in output le statistiche relative alle performance per ogni testo del set, e quelle globali per il set intero. Nel default mode, vengono fornite anche informazioni circa quali numeri rappresentativi sono aumentati o diminuiti in relazione al set annotato. Il set processato può essere aggiornato in qualsiasi momento rilanciando lo strumento in generation mode con le ultime versioni delle risorse di sistema. Inoltre, il sistema può essere eseguito in verbose mode, dove per ogni numero rappresentativo P (precision) e R (recall) al di sotto di una soglia (impostata dall'utente), verranno mostrate le annotazioni non coestensive (e i testi corrispondenti). L'output del tool è scritto in un file HTML in forma tabulare, per una facile visualizzazione dei risultati, come mostra la figura seguente.

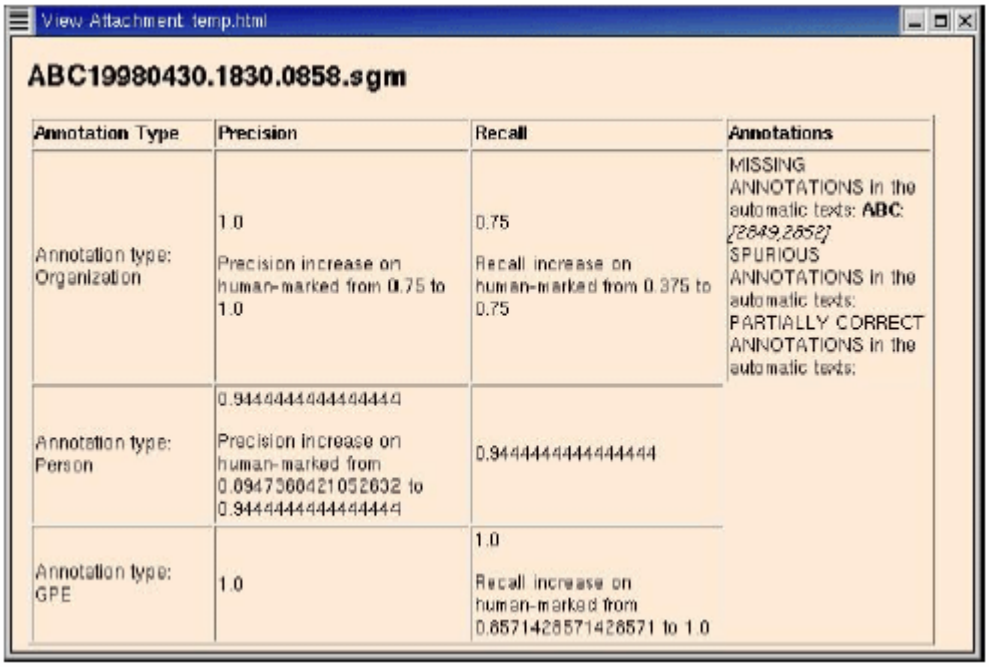


Figura 11. Frammento dei risultati del Benchmarking Tool

Le valutazioni correnti per sistemi NE MUSE eseguiti su una selezione di testi differenti producono numeri rappresentativi medi intorno al 90-95% relativi a Precision e Recall. Il sistema ANNIE di default eseguito su notizie produce numeri rappresentativi tra l'80% e il 90% di Precision e Recall. Questi numeri sono inferiori rispetto a quelli del progetto MUSE a causa della non sintonizzazione delle risorse su uno specifico tipo di testo o applicazione, ma ciò può essere fatto se necessario. I lavori relativi alla risoluzione anaforica attualmente rendono intorno al 63% per Precision e 45% per Recall, sebbene siano costantemente in progresso, e ci si aspetta che questi numeri migliorino nel prossimo futuro.

3.7.4 Metriche per la valutazione delle prestazioni nell'IE

Molte delle ricerche degli ultimi dieci anni sono state connesse alle competizioni MUC, e quindi non sorprende che siano le metriche MUC relative a precision, recall e F-measure che si tende ad usare, insieme a piccole variazioni. Di seguito si forniscono le descrizioni di queste metriche:

Precision misura il numero di elementi correttamente identificati come percentuale del numero di elementi identificati. In altre parole, misura quanti degli elementi che il sistema ha identificato sono effettivamente corretti, senza guardare gli elementi corretti che non è riuscito a recuperare. Maggiore è la precision, migliore è il sistema nell'assicurare che quello che si è identificato è corretto.
Nella classificazione binaria la precisione è analoga al valore positivo di previsione. La precisione può anche essere valutata a rispetto a un certo valore soglia, indicato con P@n, piuttosto che relativamente a tutti gli elementi identificati: in questo modo, si può valutare quanti fra i primi n elementi identificati sono rilevanti per la query.

Error rate è l'inverso delle precision, e misura il numero degli elementi non correttamente identificati come percentuale degli elementi identificati. Viene talvolta usato in alternativa alla precision.

Recall misura il numero degli elementi correttamente identificati come percentuale del numero totale di elementi corretti. In altre parole, misura quanti degli elementi che sarebbero potuti essere identificati effettivamente lo sono stati, senza guardare quante identificazioni spurie sono state fatte. Più alto è il valore di recall, migliore è il sistema nel non mancare di identificare gli elementi corretti.
Nella classificazione binaria, questo valore è chiamato sensitività.

Chiaramente, ci dovrebbe essere un bilanciamento tra precision e recall, perché un sistema può facilmente essere fatto per raggiungere il 100% di precision non identificando niente (e quindi non commettendo errori su quello che ha identificato) o il 100% di recall identificando tutto (cioè non mancando nulla). La **F-measure** [van Rijsbergen 79] è spesso usata insieme alla precision e recall, come media pesata delle due. I **False positives** sono una pratica metrica quando si ha a che fare con una vasta varietà di tipi di testo, perché non è dipendente dalla *ricchezza relativa del documento* nello stesso modo della precision. Grazie a questo noi capiamo il numero relativo delle entità di ogni tipo che possono essere trovate in un set di documenti.
Quando confrontiamo sistemi differenti sullo stesso set di documenti, la ricchezza relativa del documento diventa irrilevante, poiché è uguale per tutti i sistemi. Quando invece confrontiamo le prestazioni di un singolo sistema su diversi documenti, è molto più critico, perché se un particolare tipo di documento presenta un numero significativo di ogni tipo di entità, il risultato per quel tipo di entità può diventare distorto. Confrontiamo l'impatto sulla precision di un errore quando il numero totale delle entità corrette è uguale a 1 e quando invece è uguale a 100. Assumendo la stessa lunghezza del documento, si nota che il valore dei false positives è identico.
Le metriche comuni per la valutazione dei sistemi IE sono definite come segue:

$$\text{Precision} = \frac{\text{Correct} + 1/2\text{Partial}}{\text{Correct} + \text{Spurious} + 1/2\text{Partial}}$$

$$\text{Recall} = \frac{\text{Correct} + 1/2\text{Partial}}{\text{Correct} + \text{Missing} + 1/2\text{Partial}}$$

$$\text{F} - \text{measure} = \frac{(\beta^2 + 1)\text{P} * \text{R}}{(\beta^2\text{R}) + \text{P}}$$

dove β riflette il peso di P vs R. Se β è uguale a 1, i due hanno lo stesso peso.

$$\text{False Positive} = \frac{\text{Spurious}}{\text{c}}$$

dove c è qualche costante indipendente dalla ricchezza del documento, per esempio il numero di token o sentence nel documento.
Da notare che si considerano le annotazioni parzialmente corrette quando il tipo di entità è corretto e i frammenti sono sovrapposti ma non identici. Alle responses parzialmente corrette viene normalmente assegnato un peso medio.

Capitolo 4
APPLICAZIONE SPERIMENTALE

4.1) Descrizione dell'attività

In questo capitolo si procede alla descrizione dell'attività sperimentale svolta utilizzando GATE versione 3.1. Vengono messe in atto le nozioni logiche fornite nei capitoli precedenti, evidenziando il lato pratico dell'utilizzo del programma.
Come documenti sui quali effettuare la Named entity recognition, vengono utilizzati alcuni siti web relativi all'attività alberghiera utilizzati nel progetto WISDOM portato avanti dalla professoressa Bergamaschi come DBGROUP. L'utilità di GATE nell'analisi di questo genere di siti diventa evidente nel momento in cui l'interesse dell'utente è quello di focalizzare l'attenzione solo sui nomi di località oppure sui prezzi relativi ai vari servizi: infatti l'aiuto fornito dal programma permette di evitare lo sforzo di una lettura completa del sito.
I siti verranno analizzati in lingua inglese, poiché GATE viene fornito con l'English Tokeniser; il corrispondente Tokeniser italiano esiste ma non è si è riusciti a reperirlo in rete. Comunque si prova ad analizzare gli stessi siti in italiano effettuando un confronto tra le due resolution tramite uno dei due strumenti messi a disposizione da GATE per la valutazione delle prestazioni: l'AnnotationDiff Tool.

4.2) www.booking.com

Il primo sito che verrà "passato al setaccio" tramite GATE è www.booking.com, in lingua inglese, di cui viene fornita una snapshot di seguito:

Figura 12. www.booking.com

L'attività inizia con l'avviare GATE. All'avvio, nessun modulo o attività sono caricati: spetta all'utente farlo, indirizzando la scelta su ANNIE. Con tale modulo, in automatico vengono caricate le seguenti Processing Resources[13], rispettando tale ordine: DocumentReset, English Tokeniser, Gazetteer, Sentence Splitter, POS Tagger, NE Transducer, OrthoMatcher. Fatto ciò, essendo l'ambiente grafico (Visual Resources) caricato di default, del set CREOLE manca solo la Language Resource, ovvero il documento. Per inserirlo, è predisposta l'omonima voce nel menù, che richiede l'URL della risorsa da inserire e il tipo di codifica da utilizzare per essa: il valore di quest'ultimo parametro deve essere UTF-8, per essere in sintonia con Unicode. Ultimata questa operazione, bisogna passare il documento come parametro alle varie Processing Resources, in modo tale da creare una pipeline: da notare che l'ordine di inserimento delle risorse nella pipeline deve essere strettamente quello specificato in precedenza, pena una non corretta analisi del documento. L'ultima operazione da svolgere è quella di eseguire la pipeline, che analizzerà il documento. La parte preliminare di analisi è terminata, tutto quello che resta da fare è sottoporre al programma alcune query relative alle entità presenti nel documento. Dopo aver visualizzato il documento grazie all'editor grafico, interagendo con il pulsante *Annotation Sets* si possono vedere sulla parte destra la varie categorie relative alle entità: inserendo il segno di spunta in queste, le corrispondenti entità nel testo vengono evidenziate con lo stesso colore della categoria. Nella parte immediatamente superiore al testo, se attivata, vengono visualizzate le annotazioni relative alle entità trovate. Di seguito forniamo una snapshot di come si presenta il programma dopo aver eseguito tutte queste operazioni ed aver "spuntato" la categoria *Money*:

[13] Sono le Processing Resources descritte nel paragrafo dedicato 3.4 e sottoparagrafi.

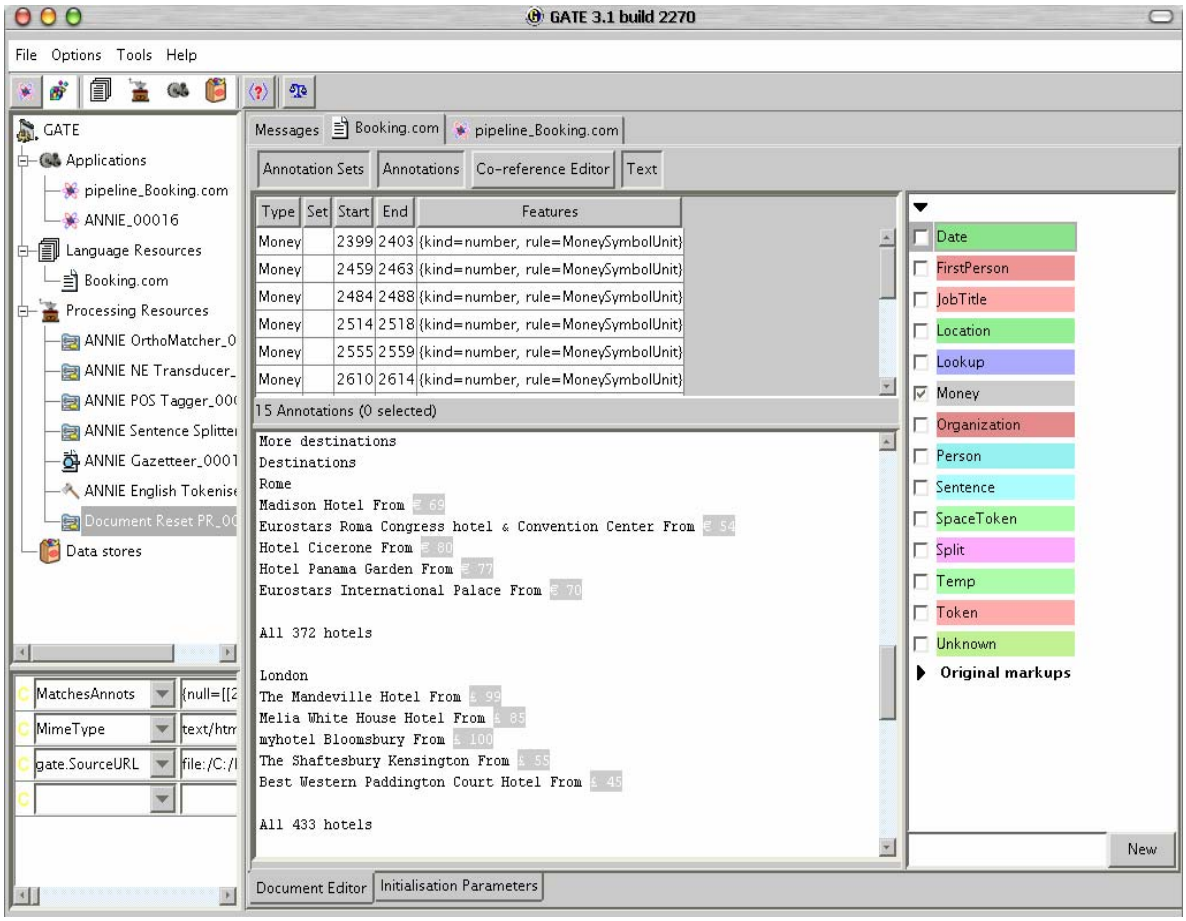


Figura 13. Analisi di www.booking.com tramite GATE

Per capire meglio come lavorano le varie Processing Resources, salviamo il documento come XML, ottenendo il codice prodotto da tali Resources e ci muoviamo tra le righe del codice alla ricerca di qualcosa di interessante.
Il codice inizia con le seguenti righe:

```
− <GateDocumentFeatures>
  − <Feature>
      <Name className="java.lang.String">gate.SourceURL</Name>
    − <Value className="java.lang.String">
        file:/C:/Documents and Settings/ferrari/Documenti/appunti/tesi/siti/booking.htm
      </Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">MimeType</Name>
      <Value className="java.lang.String">text/html</Value>
    </Feature>
  </GateDocumentFeatures>
```

che sono relative al documento; indicano che il documento da caricare presenta come URL *C:/Documents and Settings/ferrari/Documenti/appunti/tesi/siti/booking.htm* e che il tipo Mime (cioè la classificazione del file) è *text/html*.

Le linee seguenti sono relative all'attività preliminare del Tokeniser, che separa il testo in *serialized nodes* contenenti i token[14], delimitandoli con il nodo iniziale e quello finale. Riportiamo alcune linee del codice, ma deve essere ben chiaro che questa attività viene svolta per ogni token presente nel testo, il che comporta centinaia e centinaia di righe di codice:

```
− <TextWithNodes>
    <Node id="0"/>
    BOOKING
    <Node id="7"/>
    .
    <Node id="8"/>
    COM
    <Node id="11"/>
    <Node id="12"/>
    online
    <Node id="18"/>
    <Node id="19"/>
    hotel
    <Node id="24"/>
    <Node id="25"/>
    reservations
    <Node id="37"/>
    (…)
  </TextWithNodes>
```

Il numero presente nel tag *Node id* rappresenta la posizione nel testo del carattere iniziale o finale (a seconda che sia il tag di inizio o di chiusura) del token contenuto. Questi numeri sono molto importanti perché, nel prosieguo del codice, sarà a questi che faranno riferimento le annotazioni, e sarà tramite questi che noi leggendo il codice potremo risalire dall'annotazione al token.
Terminata questa operazione preliminare, il Tokeniser procede con la creazione dell'*annotation set* di default; ciò avviene percorrendo i nodi serializzati precedenti ed applicando le rules tipiche di questo modulo: rules che nella maggior parte dei casi conducono ad un confronto del token con le lists contenute nel modulo Gazetteer. Questo permette di approfondire l'analisi aggiungendo le caratteristiche intrinseche del token che, ricucendo il tutto, consentono l'effettiva risoluzione delle entità. Procediamo con l'analisi di alcuni di questi frammenti:

```
− <Annotation Id="1998" Type="SpaceToken" StartNode="3645" EndNode="3646">
  − <Feature>
      <Name className="java.lang.String">kind</Name>
      <Value className="java.lang.String">space</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">length</Name>
      <Value className="java.lang.String">1</Value>
    </Feature>
```

[14] Per il significato di token si rimanda al paragrafo 3.4.1.2

```
− <Feature>
    <Name className="java.lang.String">string</Name>
    <Value className="java.lang.String"> </Value>
  </Feature>
</Annotation>
```

La prima linea specifica il numero identificativo dell'annotazione (*Annotation Id="1998"*) ed informa che il token contenuto tra lo *StartNode="3645"* e l' *EndNode="3646"* è di tipo *"SpaceToken"*. Come detto in precedenza (paragrafo 3.4.1.2), il token SpaceToken prevede due definizioni interne, *space* e *control*. E infatti, non ci meravigliamo se le linee successive si occupano di definire proprio di quale dei due si tratti (space). Gli attributi successivi specificano la lunghezza (1) e riportano la stringa del token ( ).
Procediamo con un'altra annotazione:

```
− <Annotation Id="1688" Type="Token" StartNode="2771" EndNode="2777">
 − <Feature>
    <Name className="java.lang.String">category</Name>
    <Value className="java.lang.String">NNS</Value>
  </Feature>
 − <Feature>
    <Name className="java.lang.String">kind</Name>
    <Value className="java.lang.String">word</Value>
  </Feature>
 − <Feature>
    <Name className="java.lang.String">orth</Name>
    <Value className="java.lang.String">lowercase</Value>
  </Feature>
− <Feature>
    <Name className="java.lang.String">string</Name>
    <Value className="java.lang.String">hotels</Value>
  </Feature>
 − <Feature>
    <Name className="java.lang.String">length</Name>
    <Value className="java.lang.String">6</Value>
  </Feature>
</Annotation>
```

Il modo di interpretare la prima linea è esattamente uguale al precedente: si tratta di un'annotazione di tipo *Token*, contenuta tra i nodi *2771* e *2777*. Con queste informazioni, poche ma precise, possiamo andare a vedere di quale token effettivamente si sta parlando: basta tornare nella sezione
del codice dedicata ai serialized nodes e cercare il tag *Node id=2771*

```
<Node id="2771"/>
hotels
<Node id="2777"/>
```

cioè, il token di cui stiamo parlando è la parola *hotels*. Comunque, non c'è bisogno di effettuare questi jump tra il codice, perché le caratteristiche seguenti contengono tutte le informazioni necessarie ad una completa resolution, come si descrive adesso:

la prima feature viene inserita dal Part-of-Speech tagger, ed indica che la parola è un nome plurale (*NNS*), secondo la codifica del *Penn TreeBank Tagset* di cui riportiamo la tabella:

Penn Treebank Tagset

CC	Coordinating conjunction **e.g.** and,but,or...
CD	Cardinal Number
DT	Determiner
EX	Existential *there*
FW	Foreign Word
IN	Preposision or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List Item Marker
MD	Modal **e.g.** can, could, might, may...
NN	Noun, singular or mass
NNP	Proper Noun, singular
NNPS	Proper Noun, plural
NNS	Noun, plural
PDT	Predeterminer **e.g.** all, both ... when they precede an article
POS	Possessive Ending **e.g.** Nouns ending in 's
PRP	Personal Pronoun **e.g.** I, me, you, he...
PRP$	Possessive Pronoun **e.g.** my, your, mine, yours...
RB	Adverb Most words that end in -ly as well as degree words like quite, too and very
RBR	Adverb, comparative Adverbs with the comparative ending -er, with a strictly comparative meaning.
RBS	Adverb, superlative
RP	Particle
SYM	Symbol Should be used for mathematical, scientific or technical symbols
TO	*to*
UH	Interjection **e.g.** uh, well, yes, my...
VB	Verb, base form subsumes imperatives, infinitives and subjunctives
VBD	Verb, past tense includes the conditional form of the verb to be
VBG	Verb, gerund or persent participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner **e.g.** which, and *that* when it is used as a relative pronoun
WP	Wh-pronoun **e.g.** what, who, whom...
WP$	Possessive wh-pronoun **e.g.**
WRB	Wh-adverb **e.g.** how, where why

Figura 14. Penn Treebank Tagset

La seconda caratteristica ci rivela che il sottotipo è *word*, che implica la specifica di una ulteriore caratteristica, di tipo *orth*: in questo caso, il valore *lowercase* indica che tutte le lettere sono minuscole. La quarta feature fornisce il valore della parola, cioè la stringa, *hotels*, mentre l'ultima caratteristica è la lunghezza della parola, cioè il numero delle lettere che la compongono (*6*).
L'annotazione successiva è

```
− <Annotation Id="2578" Type="Date" StartNode="1079" EndNode="1082">
  − <Feature>
      <Name className="java.lang.String">rule1</Name>
      <Value className="java.lang.String">GazDate</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">rule2</Name>
      <Value className="java.lang.String">DateOnlyFinal</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">kind</Name>
      <Value className="java.lang.String">date</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">matches</Name>
      <Value className="java.util.ArrayList"

itemClassName="java.lang.Integer">2640;2633;2578;2626;2592;2619;2599;2585</Valu
e>
    </Feature>
</Annotation>
```

Questa volta il tipo dell'annotazione è *Date*, ed è compresa tra i nodi *1079* e *1082*. Andando a vedere tra le features, ci accorgiamo che non è presente quella che identifica la stringa: possiamo facilmente risalire a tale valore andando tra i *serialized nodes* con le coordinate *1079* e *1082*

```
<Node id="1079"/>
Mon
<Node id="1082"/>
```

la parola che si sta analizzando è dunque *Mon*.
La prima feature di questa annotazione indica che il token in gioco ha trovato riscontro in una lista del Gazetteer, più precisamente in day.lst. Questa caratteristica e la successiva rappresentano informazioni utili per la formula grammaticale scritta in linguaggio Jape per identificare entità di questo tipo. La terza feature valuta il tipo di token, aggiungendo il valore *date*. Concentriamoci adesso sull'ultima caratteristica: essa viene generata dal modulo OrthoMatcher, e rappresenta i matches riscontrati nel testo di questa entità, ovvero le varie occorrenze dell'istanza. I numeri *2640;2633;2578;2626;2592; 2619;2599;2585* sono gli Ids delle annotazioni relative alle altre occorrenze, compresa la stessa: infatti, per esempio, l'annotazione 2640 è

```
− <Annotation Id="2640" Type="Date" StartNode="1570" EndNode="1573">
  − <Feature>
```

<Name className="java.lang.String">rule1</Name>
<Value className="java.lang.String">GazDate</Value>
</Feature>
− <Feature>
<Name className="java.lang.String">rule2</Name>
<Value className="java.lang.String">DateOnlyFinal</Value>
</Feature>
− <Feature>
<Name className="java.lang.String">kind</Name>
<Value className="java.lang.String">date</Value>
</Feature>
− <Feature>
<Name className="java.lang.String">matches</Name>
<Value className="java.util.ArrayList"

itemClassName="java.lang.Integer">2640;2633;2578;2626;2592;2619;2599;2585</Valu
e>
</Feature>
</Annotation>

con

<Node id="1570"/>
Mon
<Node id="1573"/>

che, come effettivamente possiamo notare, rappresenta la stessa entità, solo diversamente disposta nel documento. Correlate ad ognuna di queste annotazioni, ce ne sono altre di questo tipo

− <Annotation Id="2228" Type="Lookup" StartNode="1079" EndNode="1082">
− <Feature>
<Name className="java.lang.String">majorType</Name>
<Value className="java.lang.String">date</Value>
</Feature>
− <Feature>
<Name className="java.lang.String">minorType</Name>
<Value className="java.lang.String">day</Value>
</Feature>
</Annotation>

Se noi ritorniamo alla "teoria" relativa al Gazetteer (paragrafo 3.4.2), troviamo che per ogni lista del Gazetteer deve essere specificato un tipo principale ed un tipo minore: per il file day.lst il tipo principale è *date* ed il tipo minore è *day*, come denotato dalla stringa *day.lst:date:day* contenuta nell'*index file*; questa annotazione altro non è che la traduzione xml di tale direttiva (*majorType=date* , *minorType=day*).
L'annotazione successiva di cui ci interessiamo è

− <Annotation Id="2559" Type="Money" StartNode="2757" EndNode="2761">
−<Feature>
<Name className="java.lang.String">rule</Name>

```
        <Value className="java.lang.String">MoneySymbolUnit</Value>
    </Feature>
  − <Feature>
        <Name className="java.lang.String">kind</Name>
        <Value className="java.lang.String">number</Value>
    </Feature>
  </Annotation>
```

Come sempre, viene specificato il numero (*2559*) ed il tipo dell'annotazione (*Money*): dunque, questa è una di quelle annotazioni il cui output è stato riportato in figura 13. Iniziando l'analisi delle caratteristiche, si nota che non sono presenti informazioni relative alla stringa dell'entità in questione: avendo ormai acquisito l'abilità, risaliamo a tale informazione andando tra i *serialized nodes* con le coordinate *2757* e *2761*; ciò che ne risulta è

```
<Node id="2757"/>
£
<Node id="2758"/>
<Node id="2759"/>
45
<Node id="2761"/>
```

che rende più chiara la comprensione delle features. Infatti il valore della prima caratteristica (*MoneySymbolUnit*) si riferisce alla partecipazione del simbolo della sterlina (*£*) mentre la seconda feature (*kind=number*) rileva l'esistenza del numero *45* subito dopo il simbolo della moneta. Procediamo con l'annotazione

```
− <Annotation Id="2567" Type="Location" StartNode="450" EndNode="456">
  − <Feature>
        <Name className="java.lang.String">rule1</Name>
        <Value className="java.lang.String">Location1</Value>
    </Feature>
  − <Feature>
        <Name className="java.lang.String">rule2</Name>
        <Value className="java.lang.String">LocFinal</Value>
    </Feature>
  − <Feature>
        <Name className="java.lang.String">locType</Name>
        <Value className="java.lang.String">city</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">matches</Name>
      <Value className="java.util.ArrayList"
      itemClassName="java.lang.Integer">2567;2568;2686</Value>
    </Feature>
  </Annotation>
```

Questa volta l'annotazione 2567, relativa all'entità compresa tra i nodi *450* e *456*, è di tipo *Location*; anche qui, non è presente tra le caratteristiche quella relativa al valore della stringa dell'entità: andiamo a procurarcela sempre tra i *serialized nodes*; il codice corrispondente è

<Node id="450"/>
Berlin
<Node id="456"/>

che comunica in modo molto limpido che il valore della stringa è *Berlin*. Come per il caso *Date*, la prime due caratteristiche forniscono informazioni relative al riscontro avvenuto con una lista del Gazetteer, che andando ad esplorare risulta essere city.lst. La terza feature restringe il campo delle località, fornendo il tipo *city*; l'ultima caratteristica è quella generata dall'OrthoMatcher, che specifica le altre occorrenze della stessa entità nel testo; ancora una volta, andiamo a verificare l'esattezza di tali matches riportando una delle altre annotazioni riportate (2686):

```
− <Annotation Id="2686" Type="Location" StartNode="465" EndNode="471">
  − <Feature>
      <Name className="java.lang.String">rule1</Name>
      <Value className="java.lang.String">Location1</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">rule2</Name>
      <Value className="java.lang.String">LocFinal</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">locType</Name>
      <Value className="java.lang.String">city</Value>
    </Feature>
 − <Feature>
     <Name className="java.lang.String">matches</Name>
     <Value className="java.util.ArrayList"
     itemClassName="java.lang.Integer">2567;2568;2686</Value>
    </Feature>
 </Annotation>
```

con

<Node id="465"/>
Berlin
<Node id="471"/>

che effettivamente realizza un match con l'entità dell'annotation 2567.
Siccome è avvenuto un riscontro con il Gazetteer, deve essere presente un'annotazione di Lookup che specifichi il tipo principale e il tipo minore della lista nella quale è contenuta la stringa del riscontro; tale annotazione è

```
− <Annotation Id="2317" Type="Lookup" StartNode="450" EndNode="456">
  − <Feature>
      <Name className="java.lang.String">majorType</Name>
      <Value className="java.lang.String">location</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">minorType</Name>
```

```
        <Value className="java.lang.String">city</Value>
    </Feature>
</Annotation>
```

la quale precisa che il tipo principale (*majorType*) è *location* e che il tipo minore (*minorType*) è *city*, in accordo con la grammatica della lista *city.lst:location:city* contenuta nell'*index file*.
Terminato l'annotation set di default, il documento xml procede con il named annotation set, cioè crea annotazioni relative alla struttura del documento in base al tipo *mime*, creando l'insieme Original Markups: queste sono per esempio immagini, paragrafi, etichette…

4.2.1 Valutazione delle prestazioni

Si procede adesso ad effettuare una valutazione dell'efficienza di GATE nella Named Entity recognition. Tale valutazione utilizza l'AnnotationDiff Tool (paragrafo 3.6.1) fornito dal sistema ed avviene in tre modi diversi:

1) confronto tra le annotazioni di GATE su www.booking.com in lingua inglese e su www.booking.com in lingua italiana;

2) confronto tra www.booking.com in inglese referenziato (con le entità annotate a mano) e www.booking.com in inglese annotato da GATE;

3) confronto tra www.booking.com in italiano referenziato (con le entità annotate a mano) e www.booking.com in italiano annotato da GATE.

1) Inglese vs Italiano
Nonostante il GATE usato per questa attività non sia predisposto per la resolution in lingua italiana, proviamo lo stesso ad utilizzarlo per tale task. Carichiamo su GATE il sito www.booking.com sia in lingua inglese che in lingua italiana, e creiamo le rispettive pipeline. Una volta eseguito ciò, selezionando il tasto *AnnotationDiff Tool* ci viene mostrata la seguente schermata nella quale abbiamo selezionato come *key document* la versione inglese del sito, come *response document* la versione italiana e come *Annotation Type* la categoria *Location*:

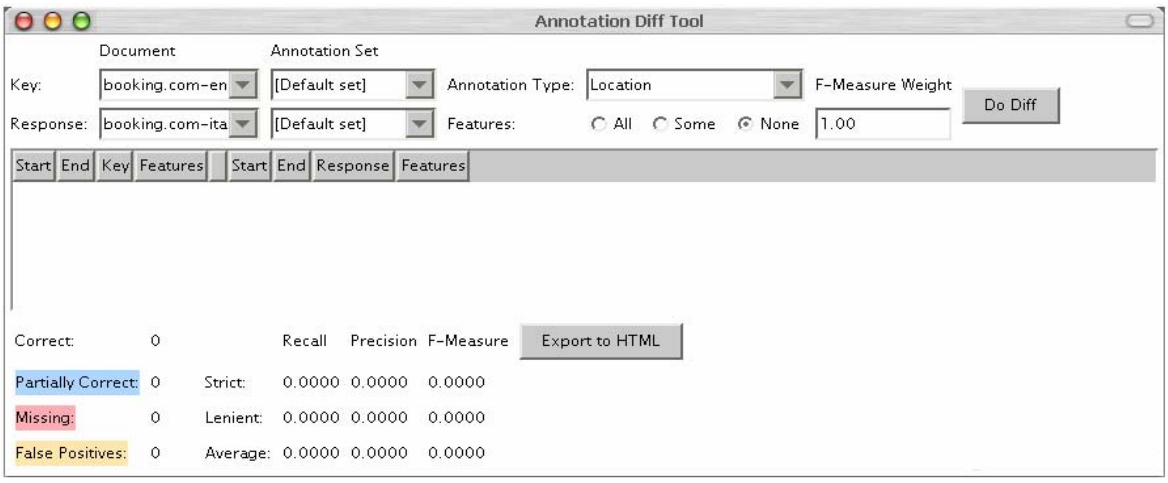


Figura 15. AnnotationDiff Tool

Cliccando sul pulsante *Do Diff*, quello che otteniamo è la schermata riportata in figura 16. Come si può notare, non è presente nessuna coppia corretta, nonostante ci siano città (come ad esempio Amsterdam) che in entrambe le lingue si scrivono nello stesso modo. Il perché allora della mancata associazione si trova nel paragrafo 3.6.2, che spiega l'importanza per tale strumento dell'offset di inizio e di fine dell'entità: per essere associate (in modo coestensivo o sovrapposto) due entità devono avere anche l'offset in un modo particolare (coincidente o parzialmente coincidente). Nel passaggio da una lingua ad un'altra, la tabulazione del sito cambia, seppur leggermente: tale lieve cambiamento comporta una totale discrepanza tra le due versioni.

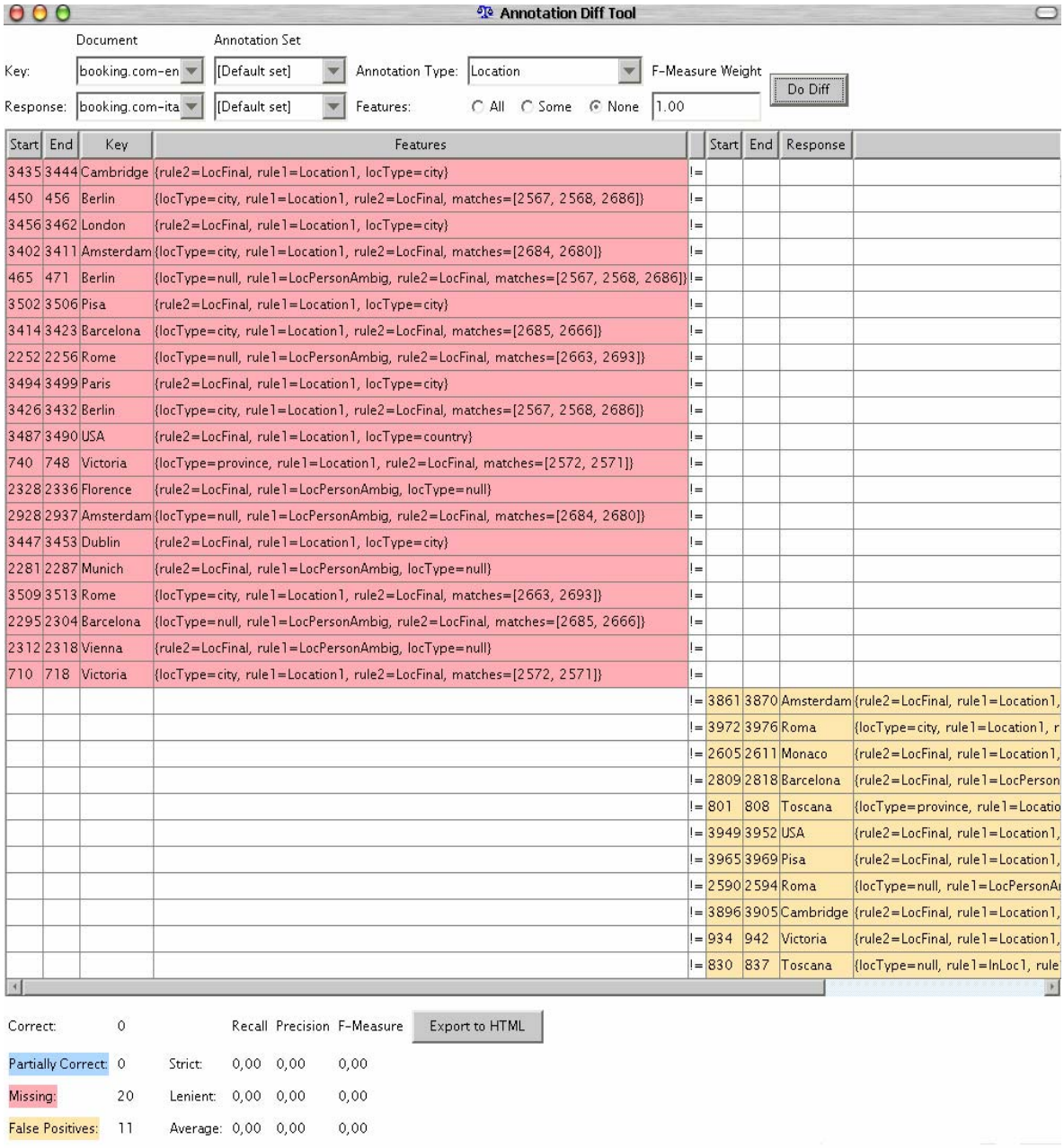


Figura 16. Risultati dell'AnnotationDiff Tool sul confronto inglese-italiano di www.booking.com

Le entità del key document vengono classificate tutte e 20 come missing mentre quelle del response document sono classificate come false positives.

2) Inglese referenziato vs Inglese

In questo tipo di valutazione, il confronto avviene tra il sito in inglese annotato manualmente (e quindi con una resolution del 100% si spera!) e il sito annotato da GATE. Carichiamo due volte su GATE il documento riguardante il sito in inglese, creiamo due pipeline e le eseguiamo: alle annotazioni generate da una delle due, però, aggiungiamo le nostre evidenziando il testo d'interesse e specificando il tipo di annotazione (*Location*). Svolta questa operazione, ci rivolgiamo all'*AnnotationDiff Tool* per la valutazione; il nostro *key document* sarà quello referenziato, il *response document* sarà quello annotato da GATE e l'*Annotation Type* sarà *Location*; il risultato è il seguente:

Start	End	Key	Features		Start	End	Response	
3487	3490	USA	{rule2=LocFinal, rule1=Location1, locType=country}	=	3487	3490	USA	{rule2=LocFinal, rule1=Location1,
2328	2336	Florence	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2328	2336	Florence	{rule2=LocFinal, rule1=LocPerson
3435	3444	Cambridge	{rule2=LocFinal, rule1=Location1, locType=city}	=	3435	3444	Cambridge	{rule2=LocFinal, rule1=Location1,
3402	3411	Amsterdam	{locType=city, rule1=Location1, rule2=LocFinal, matches=[4889, 4873, 4893]}	=	3402	3411	Amsterdam	{locType=city, rule1=Location1, r
450	456	Berlin	{locType=city, rule1=Location1, rule2=LocFinal, matches=[4777, 4895, 4776]}	=	450	456	Berlin	{locType=city, rule1=Location1, r
3509	3513	Rome	{locType=city, rule1=Location1, rule2=LocFinal, matches=[4872, 4902]}	=	3509	3513	Rome	{locType=city, rule1=Location1, r
2281	2287	Munich	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2281	2287	Munich	{rule2=LocFinal, rule1=LocPerson
3502	3506	Pisa	{rule2=LocFinal, rule1=Location1, locType=city}	=	3502	3506	Pisa	{rule2=LocFinal, rule1=Location1,
740	748	Victoria	{locType=province, rule1=Location1, rule2=LocFinal, matches=[4780, 4781]}	=	740	748	Victoria	{locType=province, rule1=Locatio
2264	2273	Amsterdam	{locType=null, rule1=LocPersonAmbig, rule2=LocFinal, matches=[4889, 4873, 4893]}	=	2264	2273	Amsterdam	{locType=null, rule1=LocPersonAi
3494	3499	Paris	{rule2=LocFinal, rule1=Location1, locType=city}	=	3494	3499	Paris	{rule2=LocFinal, rule1=Location1,
3456	3462	London	{rule2=LocFinal, rule1=Location1, locType=city}	=	3456	3462	London	{rule2=LocFinal, rule1=Location1,
2928	2937	Amsterdam	{locType=null, rule1=LocPersonAmbig, rule2=LocFinal, matches=[4889, 4873, 4893]}	=	2928	2937	Amsterdam	{locType=null, rule1=LocPersonAi
2252	2256	Rome	{locType=null, rule1=LocPersonAmbig, rule2=LocFinal, matches=[4872, 4902]}	=	2252	2256	Rome	{locType=null, rule1=LocPersonAi
3447	3453	Dublin	{rule2=LocFinal, rule1=Location1, locType=city}	=	3447	3453	Dublin	{rule2=LocFinal, rule1=Location1,
3414	3423	Barcelona	{locType=city, rule1=Location1, rule2=LocFinal, matches=[4875, 4894]}	=	3414	3423	Barcelona	{locType=city, rule1=Location1, r
2295	2304	Barcelona	{locType=null, rule1=LocPersonAmbig, rule2=LocFinal, matches=[4875, 4894]}	=	2295	2304	Barcelona	{locType=null, rule1=LocPersonAi
710	718	Victoria	{locType=province, rule1=Location1, rule2=LocFinal, matches=[4780, 4781]}	=	710	718	Victoria	{locType=city, rule1=Location1, r
3426	3432	Berlin	{locType=city, rule1=Location1, rule2=LocFinal, matches=[4777, 4895, 4776]}	=	3426	3432	Berlin	{locType=city, rule1=Location1, r
465	471	Berlin	{locType=null, rule1=LocPersonAmbig, rule2=LocFinal, matches=[4777, 4895, 4776]}	=	465	471	Berlin	{locType=null, rule1=LocPersonAi
2312	2318	Vienna	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2312	2318	Vienna	{rule2=LocFinal, rule1=LocPerson
3472	3474	PT	{}	!=				
2305	2311	Berlin	{}	!=				
659	666	Tuscany	{}	!=				
3478	3485	Norwalk	{}	!=				
2288	2294	Athens	{}	!=				
2337	2342	Paris	{}	!=				
2894	2903	Amsterdam	{}	!=				
893	902	Amsterdam	{}	!=				
959	968	Amsterdam	{}	!=				
2779	2788	Amsterdam	{}	!=				
2414	2418	Roma	{}	!=				
2577	2583	London	{}	!=				
2257	2263	London	{}	!=				
2375	2379	Rome	{}	!=				
2495	2501	Panama	{}	!=				
668	673	Italy	{}	!=				
3516	3522	Vienna	{}	!=				
529	535	London	{}	!=				
2274	2280	Milano	{}	!=				
633	640	Tuscany	{}	!=				
3465	3470	Loulé	{}	!=				
2319	2327	Brussels	{}	!=				

Figura 17. Risultati dell'AnnotationDiff Tool sul confronto inglese referenziato-inglese di www.booking.com

Come si può notare, le 21 annotazioni create delle pipeline su entrambi i documenti corrispondono perfettamente, mentre le 22 annotate manualmente vengono classificate come mancanti. E' interessante evidenziare la percentuale di risoluzione del sistema per questo caso, che è "solo" del 48,84%. Come mai una così bassa percentuale, nonostante stiamo lavorando su documenti scritti nella lingua del sistema GATE di default che usiamo? Per la risposta, dobbiamo ricordare che GATE viene fornito in una versione che, per ottenere risultati accettabili, deve essere necessariamente raffinata ed adeguata agli scopi per i quali viene utilizzata: se il nostro target è la risoluzione delle Location entity, bisogna ampliare le lists opportune contenute nel Gazetteer, così come (in misura maggiore) modificare alcune regole grammaticali, il che (soprattutto la modifica delle regole grammaticali) comporta mesi e mesi di training su documenti e corpus con vari tipi di testo, altrimenti i risultati saranno sempre di questo tipo.
Non potendo modificare le regole grammaticali, essendo un lavoro al di fuori della nostra portata, proviamo a modificare le lists del Gazetteer inserendo le entità non presenti (in questo caso *Norwalk* e *Loulè* in *city.lst, Tuscany in country.lst* e *PT* in *country_cap.lst*): come volevasi dimostrare, otteniamo un leggero miglioramento delle prestazioni, passando dal 48,84% al 58,14%:

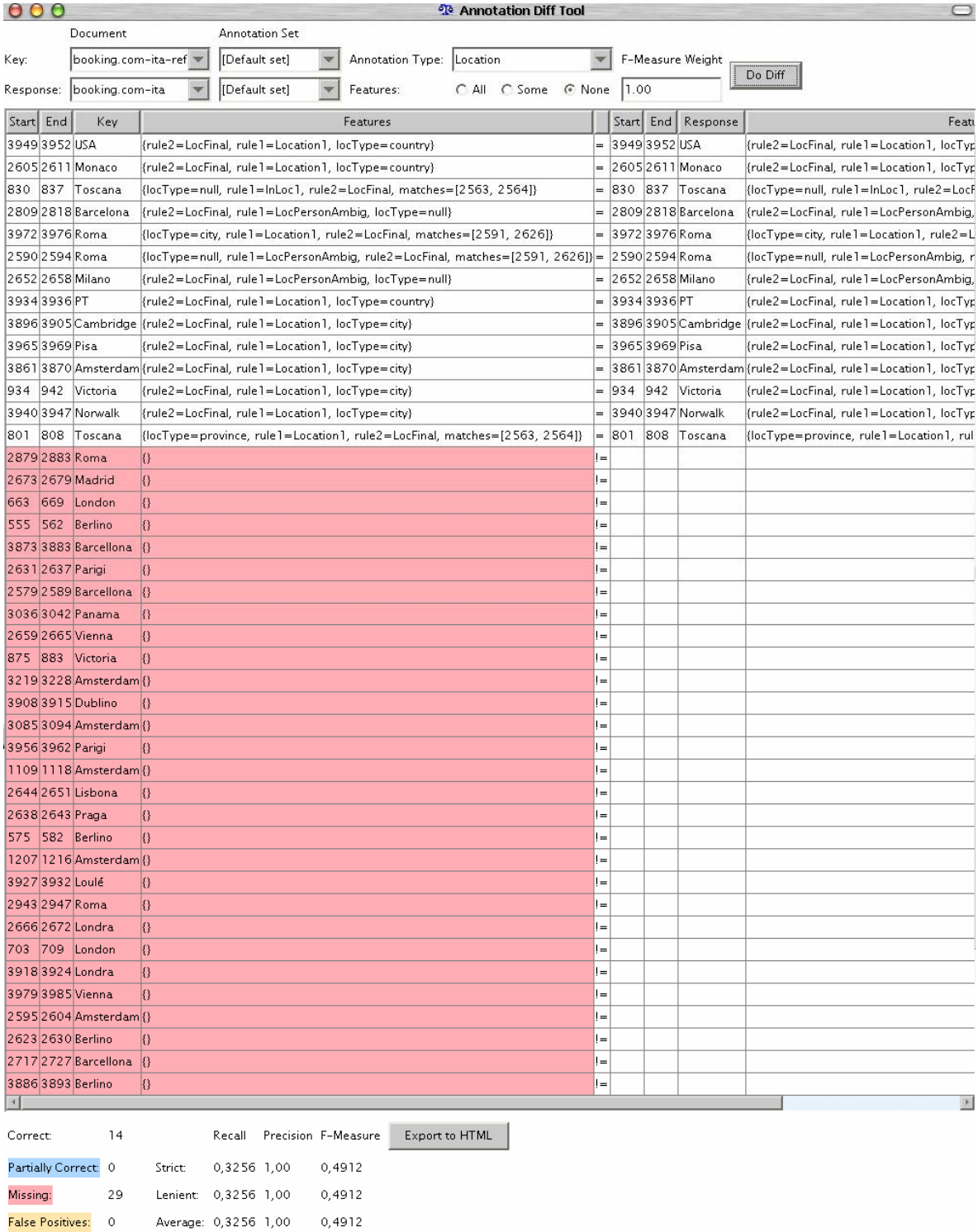
Figura 18. Risultati dell'AnnotationDiff Tool sul confronto inglese referenziato-inglese di www.booking.com dopo le modifiche

3) Italiano referenziato vs Italiano

Viene ripetuto il punto precedente con la variante della lingua: questa volta si lavora sul sito in italiano. Si può già immaginare che la percentuale di risoluzione sarà più bassa rispetto a quella relativa al sito in inglese, non essendo il sistema predisposto per la resolution in italiano.

Anche qui, carichiamo due volte il documento rappresentante il sito in italiano, creiamo le due relative pipeline e le eseguiamo: ad una delle due, aggiungiamo le entità annotate

manualmente (di tipo *Location*). Fatto ciò, lanciamo l'*AnnotationDiff Tool* sempre considerando come *key document* quello referenziato e come *response document* l'altro: il risultato è



Start	End	Key	Features		Start	End	Response	Features
3949	3952	USA	{rule2=LocFinal, rule1=Location1, locType=country}	=	3949	3952	USA	{rule2=LocFinal, rule1=Location1, locTyp
2605	2611	Monaco	{rule2=LocFinal, rule1=Location1, locType=country}	=	2605	2611	Monaco	{rule2=LocFinal, rule1=Location1, locTyp
830	837	Toscana	{locType=null, rule1=InLoc1, rule2=LocFinal, matches=[2563, 2564]}	=	830	837	Toscana	{locType=null, rule1=InLoc1, rule2=LocF
2809	2818	Barcelona	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2809	2818	Barcelona	{rule2=LocFinal, rule1=LocPersonAmbig,
3972	3976	Roma	{locType=city, rule1=Location1, rule2=LocFinal, matches=[2591, 2626]}	=	3972	3976	Roma	{locType=city, rule1=Location1, rule2=L
2590	2594	Roma	{locType=null, rule1=LocPersonAmbig, rule2=LocFinal, matches=[2591, 2626]}	=	2590	2594	Roma	{locType=null, rule1=LocPersonAmbig, r
2652	2658	Milano	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2652	2658	Milano	{rule2=LocFinal, rule1=LocPersonAmbig,
3934	3936	PT	{rule2=LocFinal, rule1=Location1, locType=country}	=	3934	3936	PT	{rule2=LocFinal, rule1=Location1, locTyp
3896	3905	Cambridge	{rule2=LocFinal, rule1=Location1, locType=city}	=	3896	3905	Cambridge	{rule2=LocFinal, rule1=Location1, locTyp
3965	3969	Pisa	{rule2=LocFinal, rule1=Location1, locType=city}	=	3965	3969	Pisa	{rule2=LocFinal, rule1=Location1, locTyp
3861	3870	Amsterdam	{rule2=LocFinal, rule1=Location1, locType=city}	=	3861	3870	Amsterdam	{rule2=LocFinal, rule1=Location1, locTyp
934	942	Victoria	{rule2=LocFinal, rule1=Location1, locType=city}	=	934	942	Victoria	{rule2=LocFinal, rule1=Location1, locTyp
3940	3947	Norwalk	{rule2=LocFinal, rule1=Location1, locType=city}	=	3940	3947	Norwalk	{rule2=LocFinal, rule1=Location1, locTyp
801	808	Toscana	{locType=province, rule1=Location1, rule2=LocFinal, matches=[2563, 2564]}	=	801	808	Toscana	{locType=province, rule1=Location1, rul
2879	2883	Roma	{}	!=				
2673	2679	Madrid	{}	!=				
663	669	London	{}	!=				
555	562	Berlino	{}	!=				
3873	3883	Barcellona	{}	!=				
2631	2637	Parigi	{}	!=				
2579	2589	Barcellona	{}	!=				
3036	3042	Panama	{}	!=				
2659	2665	Vienna	{}	!=				
875	883	Victoria	{}	!=				
3219	3228	Amsterdam	{}	!=				
3908	3915	Dublino	{}	!=				
3085	3094	Amsterdam	{}	!=				
3956	3962	Parigi	{}	!=				
1109	1118	Amsterdam	{}	!=				
2644	2651	Lisbona	{}	!=				
2638	2643	Praga	{}	!=				
575	582	Berlino	{}	!=				
1207	1216	Amsterdam	{}	!=				
3927	3932	Loulé	{}	!=				
2943	2947	Roma	{}	!=				
2666	2672	Londra	{}	!=				
703	709	London	{}	!=				
3918	3924	Londra	{}	!=				
3979	3985	Vienna	{}	!=				
2595	2604	Amsterdam	{}	!=				
2623	2630	Berlino	{}	!=				
2717	2727	Barcellona	{}	!=				
3886	3893	Berlino	{}	!=				

Figura 19. Risultati dell'AnnotationDiff Tool sul confronto italiano referenziato-italiano di www.booking.com

Solo le 14 annotazioni create su entrambi i documenti realizzano le coppie, mentre le 29 create a mano vengono classificate come missing: tutto ciò realizza una resolution pari al 32,56%. Da notare che le entry inserite nelle lists del Gazetteer nel passo precedente

(*Norwalk* e *PT*) sono state utilizzate per la resolution. Inserendo anche qui nelle lists del Gazetteer le entità mancanti (*Parigi*, *Londra*, *Praga*, *Lisbona*, *Barcellona*, *Berlino*, *Dublino* in *city.lst*), proviamo a migliorare le prestazioni:

Figura 20. Risultati dell'AnnotationDiff Tool sul confronto italiano referenziato-italiano di www.booking.com dopo le modifiche

Come si può notare, si passa dalle 14 alle 24 coppie corrette, portando la percentuale di risoluzione dal 32,56 al 55,81%.

4.3) www.venere.com

Passiamo ora ad analizzare tramite GATE un altro sito, sempre in inglese: www.venere.com di cui riportiamo la snapshot della homepage.

Figura 21. www.venere.com

Anche qui bisogna svolgere l'attività preliminare, che consente a GATE di eseguire l'analisi, che è perfettamente uguale a quella svolta per il sito precedente ma che per comodità riportiamo. Per prima cosa, bisogna avviare GATE che, all'avvio, non presenta nessun modulo caricato; per importare tali moduli, si clicca sul pulsante *Load ANNIE with default*. ANNIE verrà caricato con tutte le sue Processing Resources di default, che sono ordinate come segue: DocumentReset, English Tokeniser, Gazetteer, Sentence Splitter, POS Tagger, NE Transducer, OrthoMatcher. Avendo svolto la parte relativa alle Processing Resources, resta da caricare la Language Resource, ossia il documento. Per inserirlo, è predisposta l'omonima voce nel menù, che richiede l'URL della risorsa da inserire e il tipo di codifica da utilizzare per essa: il valore di quest'ultimo parametro deve essere UTF-8, per essere in sintonia con Unicode. Terminato ciò, bisogna realizzare una pipeline con le Processing Resources caricate in precedenza: come parametro di questa pipeline verrà passato il documento appena caricato; l'ordine delle Processing Resources all'interno della pipeline deve strettamente essere quello precedente; pipeline che una volta creata va eseguita. Ultimata questa operazione, il sistema è pronto ed in grado di analizzare il documento, restituendo le entità richieste. Grazie all'editor grafico, il documento viene visualizzato; cliccando sul pulsante *Annotation Sets* compaiono sulla destra del programma la categorie relative alle entità: selezionandone una, le corrispondenti entità nel testo

vengono evidenziate col medesimo colore. La figura presentata di seguito fornisce appunto una visione del sito caricato da GATE ed analizzato secondo la query relativa alle *Location*:

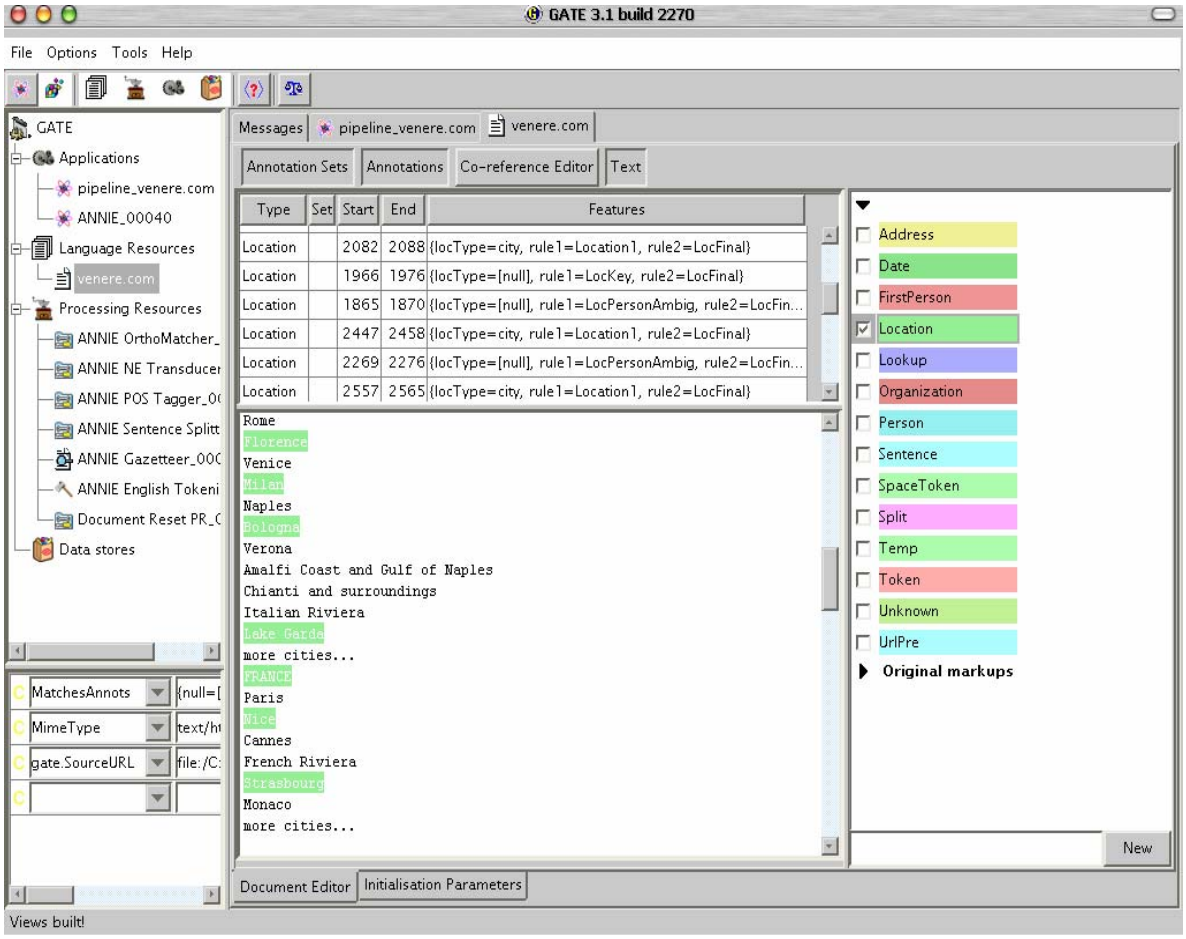


Figura 22. Analisi di www.venere.com tramite GATE

Salvando il documento analizzato come XML, possiamo procedere con un'analisi di come le Processing Resources hanno lavorato.
La prima parte del file xml è

```
− <GateDocumentFeatures>
  − <Feature>
      <Name className="java.lang.String">gate.SourceURL</Name>
    − <Value className="java.lang.String">
      file:/C:/Documents and Settings/ferrari/Documenti/appunti/tesi/siti/Venere.htm
      </Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">MimeType</Name>
      <Value className="java.lang.String">text/html</Value>
    </Feature>
 </GateDocumentFeatures>
```

che contiene alcune informazioni preliminari riguardo il file, come per esempio il suo Uniform Resource Locator (*C:/Documents and Settings/ferrari/Documenti/appunti/tesi/siti/Venere.htm*) ed il suo tipo *Mime* (*text/html*). La parte successiva del file xml è quella dei *serialized nodes*, cioè quella operazione preliminare svolta dal Tokeniser che suddivide il testo i token elementari. Uno stralcio delle linee di codice relative a questa parte è

```
− <TextWithNodes>
  <Node id="0"/>
  Hotel
  <Node id="5"/>
  <Node id="6"/>
  reservations
  <Node id="18"/>
  <Node id="19"/>
   -
  <Node id="20"/>
  <Node id="21"/>
  Venere
  <Node id="27"/>
  .
  <Node id="28"/>
  com
  <Node id="31"/>
  <Node id="32"/>
  (…)
  </TextWithNodes>
```

Come già detto, i numeri presenti nei tag come valori di *Node id* denotano la posizione del token compreso tra i due tag e sono quelli ai quali le annotazioni successive si riferiranno.
Ultimata la precedente operazione preliminare, il Tokeniser esegue la creazione dell' *annotation set* di default; questa attività viene realizzata ripercorrendo la sezione precedente dei nodi serializzati e rispettando alcune rules specifiche di tale Processing Resource, che molto spesso si riconducono ad un confronto del token con le entry delle lists del Gazetteer. Andiamo ad approfondire questi concetti analizzando frammenti di codice:

```
− <Annotation Id="928" Type="Token" StartNode="685" EndNode="690">
  − <Feature>
      <Name className="java.lang.String">category</Name>
      <Value className="java.lang.String">NNS</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">string</Name>
      <Value className="java.lang.String">Rooms</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">kind</Name>
      <Value className="java.lang.String">word</Value>
    </Feature>
```

```
  − <Feature>
      <Name className="java.lang.String">orth</Name>
      <Value className="java.lang.String">upperInitial</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">length</Name>
      <Value className="java.lang.String">5</Value>
    </Feature>
</Annotation>
```


La prima linea contiene il numero dell'annotazione (*928*), il tipo della stessa (*Token*) e le coordinate del token in questione tra i serialized nodes (*StartNode="685" EndNode="690"*). La prima feature, inserita dal *Part of Speech tagger*, fornisce una descrizione del token assegnando alla variabile *category* il valore *NNS* secondo la Penn Treebank Set: il corrispondente significato della sigla NNS è *nome plurale*. Ed in effetti, la caratteristica successiva è quella relativa alla stringa del token, che nel nostro caso ha valore *Rooms*; ciò rispetta la feature precedente, essendo realmente un nome plurale. La terza caratteristica specifica il sottotipo della categoria Token, *word*, che implica la descrizione della composizione della parola stessa (*orth*), effettuata nella feature successiva: il valore *upperInitial* significa che la parola inizia con una lettera maiuscola e procede con lettere tutte minuscole. L'ultima caratteristica elenca la lunghezza del token (*5*).
La seconda annotazione analizzata è


```
− <Annotation Id="1990" Type="SpaceToken" StartNode="219" EndNode="220">
  − <Feature>
      <Name className="java.lang.String">kind</Name>
      <Value className="java.lang.String">control</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">length</Name>
      <Value className="java.lang.String">1</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">string</Name>
      <Value className="java.lang.String"> </Value>
    </Feature>
 </Annotation>
```


Anche qui, la prima riga contiene il numero dell'annotazione (*1990*) ed i nodi tra i quali il token è contenuto (*219* e *220*); il tipo di annotazione però è *SpaceToken*. Abbiamo già visto (paragrafo 3.4.1.2) che per questa categoria è prevista una distinzione interna tra space e control: come spiega la prima caratteristica, questo caso è di tipo control. Essendo un carattere di spazio, la sua lunghezza è unitaria (seconda feature) e la stringa rappresentativa è " ".
La successiva annotazione è


```
− <Annotation Id="2210" Type="Split" StartNode="2265" EndNode="2268">
  − <Feature>
      <Name className="java.lang.String">kind</Name>
      <Value className="java.lang.String">external</Value>
```

```
    </Feature>
  </Annotation>
```

Questa annotazione viene prodotta dal *Sentence Splitter*; dovendo dividere il testo in periodi, ha bisogno di riconoscere quale punteggiatura effettivamente rappresenta la fine di una frase: nel nostro caso il codice dei serialized nodes racchiuso tra i nodi *2265* e *2268* è

```
<Node id="2265"/>
.
<Node id="2266"/>
.
<Node id="2267"/>
.
<Node id="2268"/>
```

ovvero tre puntini sospensivi, che realmente rappresentano la fine di una frase.
Il tipo *external* permette di capire che il periodo in questione rappresenta uno dei due estremi della frase di cui fa parte, in questo caso l'estremo finale; il valore internal l'avremmo incontrato qualora il carattere terminatore del periodo fosse stato ; o : .
La prossima annotazione è

```
− <Annotation Id="2356" Type="Date" StartNode="2704" EndNode="2717">
  − <Feature>
      <Name className="java.lang.String">rule1</Name>
      <Value className="java.lang.String">DateName</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">rule2</Name>
      <Value className="java.lang.String">DateOnlyFinal</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">kind</Name>
      <Value className="java.lang.String">date</Value>
    </Feature>
 </Annotation>
```

Questa annotazione (2356) è di tipo Date. I primi due attributi come al solito sono regole scritte in linguaggio *Jape* per effettuare tale tipo di risoluzione e segnalano che è avvenuto un riscontro con il Gazetteer; la prima però (*rule1*) avendo valore *DateName* ci comunica che si tratta di una data completa: non essendo presente la stringa corrispondente, andiamo a risalire ad essa tramite le coordinate dei nodi. Il corrispondente codice è

```
<Node id="2704"/>
Mon
<Node id="2707"/>
<Node id="2708"/>
11
<Node id="2710"/>
<Node id="2711"/>
Sep
<Node id="2714"/>
```

<Node id="2715"/>
06
<Node id="2717"/>

ovvero *Mon 11 Sep 06*.
Procediamo con l'annotazione

```
− <Annotation Id="2332" Type="Location" StartNode="2082" EndNode="2088">
  − <Feature>
      <Name className="java.lang.String">rule1</Name>
      <Value className="java.lang.String">Location1</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">rule2</Name>
      <Value className="java.lang.String">LocFinal</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">locType</Name>
      <Value className="java.lang.String">city</Value>
    </Feature>
  </Annotation>
```

Questa volta l'annotazione 2332, relativa all'entità compresa tra i nodi *2082* e *2088*, è di tipo *Location*; anche qui, non è presente tra le caratteristiche quella relativa al valore della stringa dell'entità: andiamo a procurarcela sempre tra i *serialized nodes*; il codice corrispondente è

```
<Node id="2082"/>
Munich
<Node id="2088"/>
```

dove, quindi, la stringa d'interesse risulta essere *Munich*. La prime due caratteristiche, come al solito nei casi di questi tipi di annotazione, sono indicative di un riscontro avvenuto con una lista del Gazetteer, e più precisamente *city.lst*. L'ultima feature raffina il tipo di token assegnandogli il valore *city*.
Essendo avvenuto un riscontro con il Gazetteer, risulta essere presente un'annotazione di tipo Lookup che specifichi il tipo principale e il tipo minore della lista nella quale è contenuta la stringa del riscontro; tale annotazione è

```
− <Annotation Id="2176" Type="Lookup" StartNode="2082" EndNode="2088">
  − <Feature>
      <Name className="java.lang.String">majorType</Name>
      <Value className="java.lang.String">location</Value>
    </Feature>
  − <Feature>
      <Name className="java.lang.String">minorType</Name>
      <Value className="java.lang.String">city</Value>
    </Feature>
  </Annotation>
```

la quale precisa che il tipo principale (*majorType*) è *location* e che il tipo minore (*minorType*) è *city*, rispettando la grammatica della lista *city.lst:location:city* contenuta nell'*index file*.

4.3.2 Valutazione delle prestazioni

Qui di seguito si procede, come avvenuto per il sito precedente, ad una valutazione delle prestazioni di GATE tramite l'AnnotationDiff Tool. Si segue la seguente scaletta di valutazione:

1) confronto tra le annotazioni di GATE su www.venere.com in lingua inglese e su www.venere.com in lingua italiana;

2) confronto tra www.venere.com in inglese referenziato (con le entità annotate a mano) e www.venere.com in inglese annotato da GATE;

3) confronto tra www.venere.com in italiano referenziato (con le entità annotate a mano) e www.venere.com in italiano annotato da GATE.

1) Inglese vs Italiano
Si effettua il confronto delle resolution di GATE del sito www.venere.com in lingua inglese ed in lingua italiana. Anche qui, valgono le stesse considerazioni fatte nello stesso punto del sito precedente, e cioè che la versione di GATE che usiamo non è predisposta alla resolution italiana. Come al solito, carichiamo i due documenti e creiamo le due pipeline associate. Una volta lanciate tali pipeline, i documenti sono pronti per permettere la visualizzazione delle entità. Possiamo verificare che tutto sia andato bene inserendo il segno di spunta in alcune categorie presenti sulla destra, visualizzandone gli elementi riscontrati nel testo. Dopodichè, per effettuare la valutazione, lanciamo l'*AnnotationDiff Tool*, considerando il documento in inglese come *key document* e quello in italiano come *response document*, oltre che *Location* come *Annotation Type*:

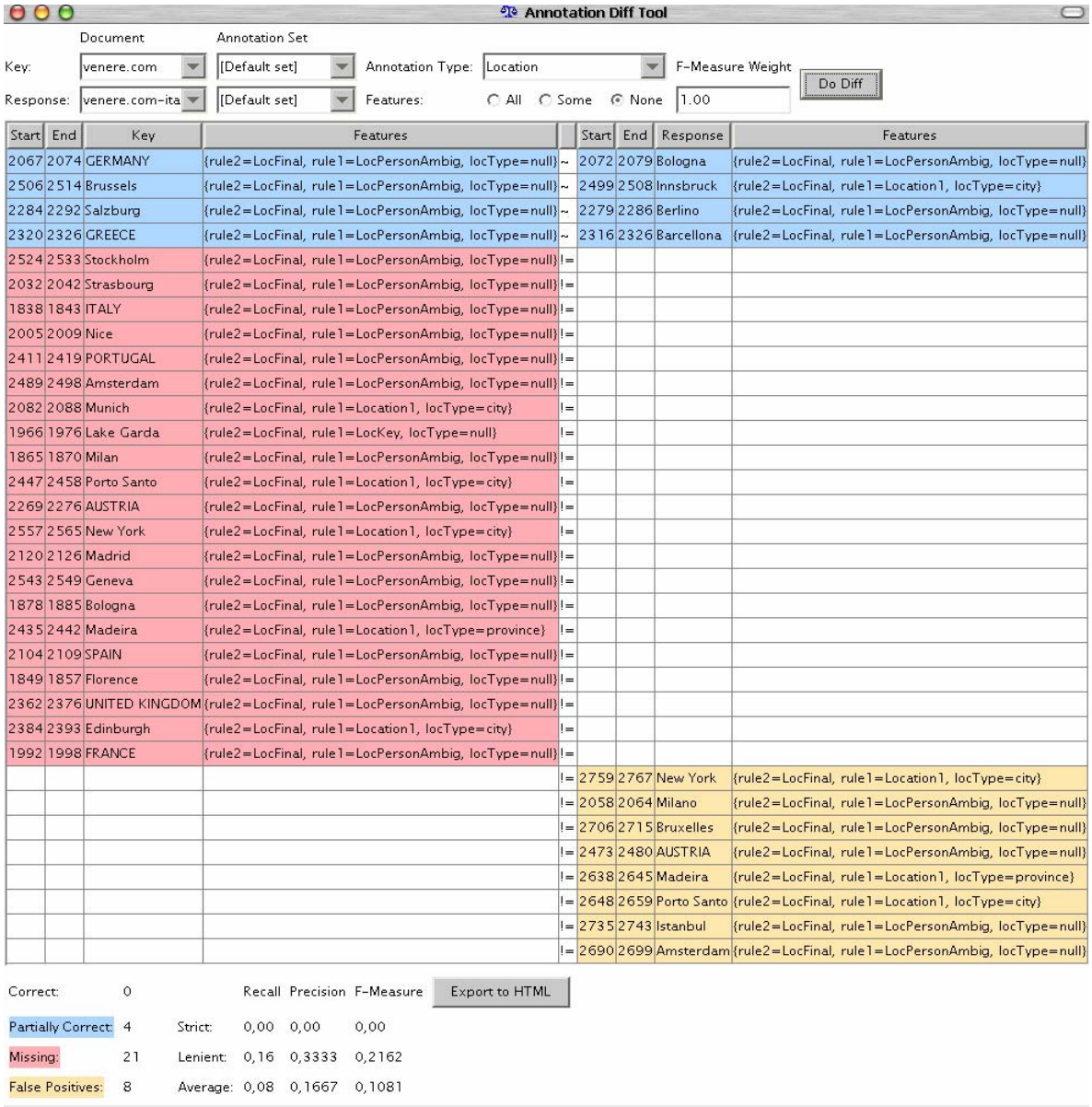


Figura 23. Risultati dell'AnnotationDiff Tool sul confronto inglese-italiano di www.venere.com

Notiamo come al solito che le entità del key document vengono classificate come missing e quelle del response document come False Positives, ma la cosa più interessante alla quale fare attenzione è la presenza di 4 coppie di entità classificate come parzialmente corrette: andando a guardare gli abbinamenti, si vede che nessuna coppia è formata da entità in qualche modo simili. Allora, il perché di tale classificazione lo si può trovare nella definizione stessa di parzialmente corretta (paragrafo 3.6.2), che riportiamo per comodità:

Due annotazioni sono parzialmente compatibili se sono sovrapposte e se le features di una (di solito quella proveniente dal key set) sono contenute nelle features dell'altra (solitamente quella proveniente dal response set).

In effetti, le due entità che formano ciascuna coppia presentano gli offset che in qualche modo sono sovrapposti, e le features che combaciano.
E' interessante notare anche come cambiano i valori di Recall, Precision e F-Measure a seconda che il metro di valutazione sia strict (considera come incorrette le entità

parzialmente corrette), lenient (considera come corrette le entità parzialmente corrette) o average (la media dei due precedenti).

2) Inglese referenziato vs Inglese

Questa valutazione avviene tra il documento in inglese annotato a mano (e quindi tendente alla massima resolution) e lo stesso documento analizzato invece dal sistema. Carichiamo quindi due volte il documento relativo al sito su GATE, e creiamo le relative pipeline. Ad uno dei due documenti, però, aggiungiamo le annotazioni create manualmente, evidenziando il testo specifico e selezionando la categoria di tale annotazione. Per effettuare la valutazione, lanciamo l'*AnnotationDiff Tool* con i seguenti parametri: il documento referenziato come *key document*, l'altro come *response document* e *Location* come *Annotation Type*. Il risultato è visibile nella figura 24. Come si può notare, le 25 entità annotate delle pipeline realizzano i matches, mentre le 31 entità annotate manualmente vengono classificate come missing. Anche qui, per migliorare la percentuale del 44,64% di risoluzione, bisogna intervenire sulle liste del Gazetteer. Infatti, apportando alcune aggiunte (*Amalfi Coast*, *Gulf of Naples*, *Chianti*, *Italian Riviera*, *Canary Islands*, *Balearic Islands*, *French Riviera* in *region.lst*, *Algarve* e *Granada* in *city.lst*, *Crete* in *country.lst*) la percentuale di resolution migliora, passando dal 44,64% al 55,36%, come si può vedere in figura 25. Da notare le coppie formate dalle entità referenziate, che non presentano nessuna feature, e dalle entità successivamente annotate, complete in tutto e per tutto di ogni caratteristica. Questo è possibile grazie alla modalità di match tra una entità di tipo unknow ed una invece dotata di tutte le informazioni.

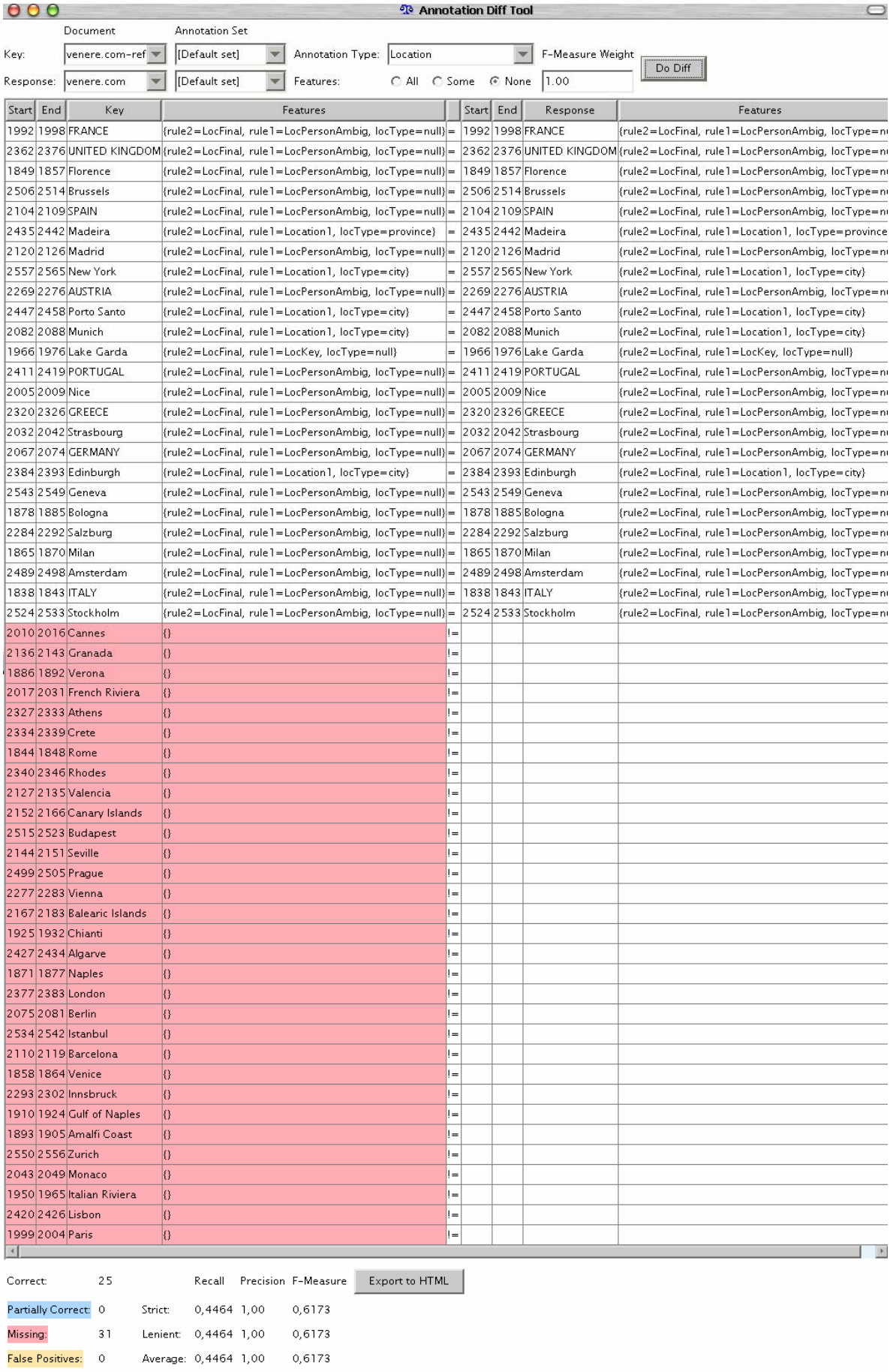


Figure 24. Risultati dell'AnnotationDiff Tool sul confronto inglese referenziato-inglese di www.venere.com

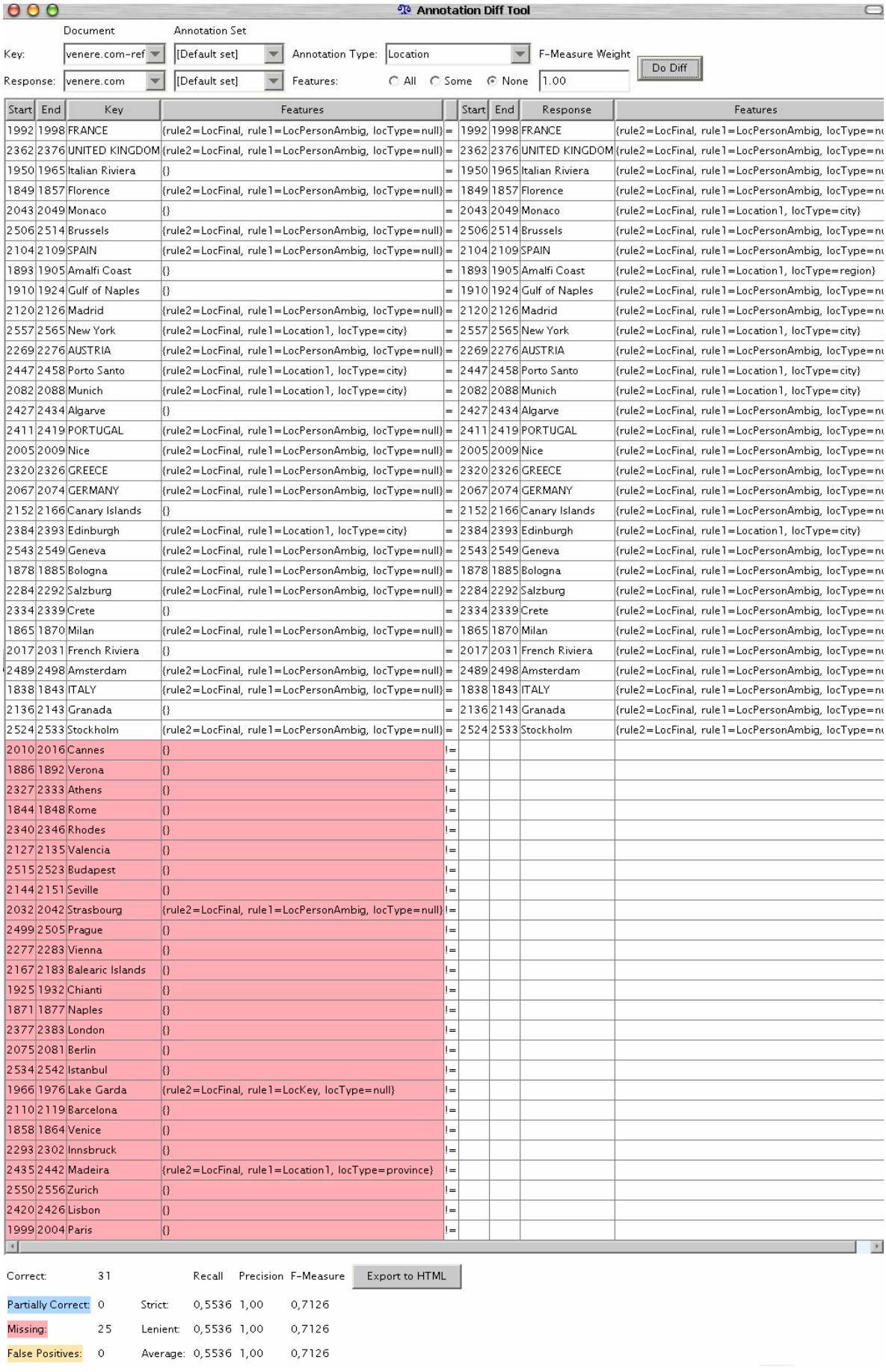
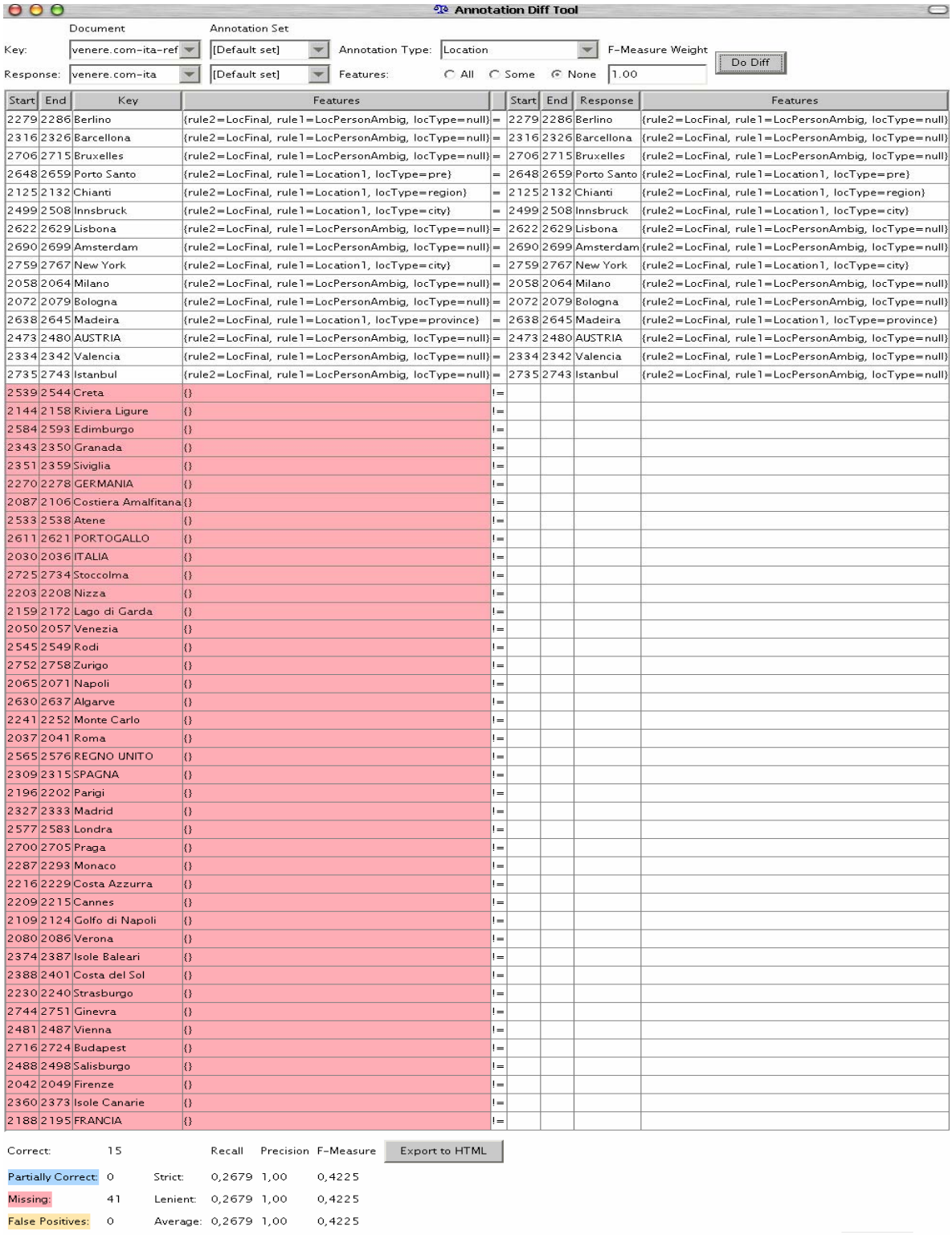


Figura 25. Risultati dell'AnnotationDiff Tool sul confronto inglese referenziato-inglese di www.venere.com dopo le modifiche

3) Italiano referenziato vs Italiano

Punto del tutto simile al precedente, eccezion fatta per la lingua del documento: questa volta si lavora con documenti in italiano. Carichiamo due volte il documento su GATE, creiamo le due pipeline e ad una di esse aggiungiamo le annotazioni manualmente. Per la valutazione lanciamo l'AnnotationDiff Tool, caratterizzato come sempre dai seguenti parametri: il documento referenziato come *key document*, l'altro come *response document* e *Location* come *Annotation Type*. Il risultato è



Start	End	Key	Features		Start	End	Response	Features
2279	2286	Berlino	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2279	2286	Berlino	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2316	2326	Barcellona	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2316	2326	Barcellona	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2706	2715	Bruxelles	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2706	2715	Bruxelles	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2648	2659	Porto Santo	{rule2=LocFinal, rule1=Location1, locType=pre}	=	2648	2659	Porto Santo	{rule2=LocFinal, rule1=Location1, locType=pre}
2125	2132	Chianti	{rule2=LocFinal, rule1=Location1, locType=region}	=	2125	2132	Chianti	{rule2=LocFinal, rule1=Location1, locType=region}
2499	2508	Innsbruck	{rule2=LocFinal, rule1=Location1, locType=city}	=	2499	2508	Innsbruck	{rule2=LocFinal, rule1=Location1, locType=city}
2622	2629	Lisbona	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2622	2629	Lisbona	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2690	2699	Amsterdam	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2690	2699	Amsterdam	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2759	2767	New York	{rule2=LocFinal, rule1=Location1, locType=city}	=	2759	2767	New York	{rule2=LocFinal, rule1=Location1, locType=city}
2058	2064	Milano	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2058	2064	Milano	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2072	2079	Bologna	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2072	2079	Bologna	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2638	2645	Madeira	{rule2=LocFinal, rule1=Location1, locType=province}	=	2638	2645	Madeira	{rule2=LocFinal, rule1=Location1, locType=province}
2473	2480	AUSTRIA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2473	2480	AUSTRIA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2334	2342	Valencia	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2334	2342	Valencia	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2735	2743	Istanbul	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2735	2743	Istanbul	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}
2539	2544	Creta	{}	!=				
2144	2158	Riviera Ligure	{}	!=				
2584	2593	Edimburgo	{}	!=				
2343	2350	Granada	{}	!=				
2351	2359	Siviglia	{}	!=				
2270	2278	GERMANIA	{}	!=				
2087	2106	Costiera Amalfitana	{}	!=				
2533	2538	Atene	{}	!=				
2611	2621	PORTOGALLO	{}	!=				
2030	2036	ITALIA	{}	!=				
2725	2734	Stoccolma	{}	!=				
2203	2208	Nizza	{}	!=				
2159	2172	Lago di Garda	{}	!=				
2050	2057	Venezia	{}	!=				
2545	2549	Rodi	{}	!=				
2752	2758	Zurigo	{}	!=				
2065	2071	Napoli	{}	!=				
2630	2637	Algarve	{}	!=				
2241	2252	Monte Carlo	{}	!=				
2037	2041	Roma	{}	!=				
2565	2576	REGNO UNITO	{}	!=				
2309	2315	SPAGNA	{}	!=				
2196	2202	Parigi	{}	!=				
2327	2333	Madrid	{}	!=				
2577	2583	Londra	{}	!=				
2700	2705	Praga	{}	!=				
2287	2293	Monaco	{}	!=				
2216	2229	Costa Azzurra	{}	!=				
2209	2215	Cannes	{}	!=				
2109	2124	Golfo di Napoli	{}	!=				
2080	2086	Verona	{}	!=				
2374	2387	Isole Baleari	{}	!=				
2388	2401	Costa del Sol	{}	!=				
2230	2240	Strasburgo	{}	!=				
2744	2751	Ginevra	{}	!=				
2481	2487	Vienna	{}	!=				
2716	2724	Budapest	{}	!=				
2488	2498	Salisburgo	{}	!=				
2042	2049	Firenze	{}	!=				
2360	2373	Isole Canarie	{}	!=				
2188	2195	FRANCIA	{}	!=				

Figura 26. Risultati dell'AnnotationDiff Tool sul confronto italiano referenziato-italiano di www.venere.com

La percentuale è molto bassa, del 26,79%. In effetti, solo 15 entità sono state annotate del sistema, mentre 41 sono quelle referenziate. Andiamo anche qui a modificare i file .lst del Gazetteer (*Edimburgo*, *Siviglia*, *Atene*, *Stoccolma*, *Nizza*, *Venezia*, *Rodi*, *Zurigo*, *Monte Carlo*, *Strasburgo*, *Ginevra*, *Salisburgo*, *Firenze* in *city.lst*, *Riviera Ligure*, *Costiera Amalfitana*, *Lago di Garda*, *Costa Azzurra*, *Golfo di Napoli*, *Costa del Sol*, *Isole Baleari*, *Isole Canarie* in *region.lst*, *FRANCIA*, *SPAGNA*, *REGNO UNITO*, *ITALIA*, *PORTOGALLO*, *GERMANIA* in *country_cap.lst*); il confronto corrispondente è

Start	End	Key	Features		Start	End	Response	Features
2188	2195	FRANCIA	{}	=	2188	2195	FRANCIA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2360	2373	Isole Canarie	{}	=	2360	2373	Isole Canarie	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2042	2049	Firenze	{}	=	2042	2049	Firenze	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2488	2498	Salisburgo	{}	=	2488	2498	Salisburgo	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2706	2715	Bruxelles	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2706	2715	Bruxelles	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2744	2751	Ginevra	{}	=	2744	2751	Ginevra	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2230	2240	Strasburgo	{}	=	2230	2240	Strasburgo	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2648	2659	Porto Santo	{rule2=LocFinal, rule1=Location1, locType=pre}	=	2648	2659	Porto Santo	{rule2=LocFinal, rule1=Location1, locType=city}
2109	2124	Golfo di Napoli	{}	=	2109	2124	Golfo di Napoli	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2287	2293	Monaco	{}	=	2287	2293	Monaco	{rule2=LocFinal, rule1=Location1, locType=countr
2690	2699	Amsterdam	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2690	2699	Amsterdam	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2327	2333	Madrid	{}	=	2327	2333	Madrid	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2309	2315	SPAGNA	{}	=	2309	2315	SPAGNA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2565	2576	REGNO UNITO	{}	=	2565	2576	REGNO UNITO	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2759	2767	New York	{rule2=LocFinal, rule1=Location1, locType=city}	=	2759	2767	New York	{rule2=LocFinal, rule1=Location1, locType=city}
2630	2637	Algarve	{}	=	2630	2637	Algarve	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2058	2064	Milano	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2058	2064	Milano	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2545	2549	Rodi	{}	=	2545	2549	Rodi	{rule2=LocFinal, rule1=Location1, locType=city}
2203	2208	Nizza	{}	=	2203	2208	Nizza	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2725	2734	Stoccolma	{}	=	2725	2734	Stoccolma	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2072	2079	Bologna	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2072	2079	Bologna	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2030	2036	ITALIA	{}	=	2030	2036	ITALIA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2611	2621	PORTOGALLO	{}	=	2611	2621	PORTOGALLO	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2087	2106	Costiera Amalfitana	{}	=	2087	2106	Costiera Amalfitana	{rule2=LocFinal, rule1=Location1, locType=region
2270	2278	GERMANIA	{}	=	2270	2278	GERMANIA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2473	2480	AUSTRIA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	=	2473	2480	AUSTRIA	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2343	2350	Granada	{}	=	2343	2350	Granada	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2584	2593	Edimburgo	{}	=	2584	2593	Edimburgo	{rule2=LocFinal, rule1=Location1, locType=city}
2144	2158	Riviera Ligure	{}	=	2144	2158	Riviera Ligure	{rule2=LocFinal, rule1=LocPersonAmbig, locType=
2539	2544	Creta	{}	!=				
2735	2743	Istanbul	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	!=				
2334	2342	Valencia	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	!=				
2351	2359	Siviglia	{}	!=				
2533	2538	Atene	{}	!=				
2638	2645	Madeira	{rule2=LocFinal, rule1=Location1, locType=province}	!=				
2159	2172	Lago di Garda	{}	!=				
2050	2057	Venezia	{}	!=				
2752	2758	Zurigo	{}	!=				
2065	2071	Napoli	{}	!=				
2241	2252	Monte Carlo	{}	!=				
2037	2041	Roma	{}	!=				
2196	2202	Parigi	{}	!=				
2577	2583	Londra	{}	!=				
2700	2705	Praga	{}	!=				
2216	2229	Costa Azzurra	{}	!=				
2622	2629	Lisbona	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	!=				
2209	2215	Cannes	{}	!=				
2499	2508	Innsbruck	{rule2=LocFinal, rule1=Location1, locType=city}	!=				
2080	2086	Verona	{}	!=				
2374	2387	Isole Baleari	{}	!=				
2388	2401	Costa del Sol	{}	!=				
2125	2132	Chianti	{rule2=LocFinal, rule1=Location1, locType=region}	!=				
2481	2487	Vienna	{}	!=				
2716	2724	Budapest	{}	!=				
2316	2326	Barcellona	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	!=				
2279	2286	Berlino	{rule2=LocFinal, rule1=LocPersonAmbig, locType=null}	!=				

Figura 27. Risultati dell'AnnotationDiff Tool sul confronto italiano referenziato-italiano di www.venere.com dopo le modifiche

Si è passati dal riscontro di 15 coppie a quello di 29, portando la percentuale di resolution dal 26,79% al 51,79%, percentuale ancora abbastanza bassa (più o meno una entità su due) ma comunque nettamente migliore della precedente. Bisogna inoltre ricordare che il sistema in questione non è configurato per realizzare al meglio la risoluzione delle entità in italiano.

Riepilogo, Conclusioni e Sviluppi futuri

Giunti al termine dell'esperienza, è giusto ricapitolare un po' tutto ed arrivare a trarre delle conclusioni. La prima parte dell'attività è stata volta ad acquisire le conoscenze teoriche relative all'Information Extraction, capire gli ambiti del suo impiego ed i problemi per la cui risoluzione è preposto. E' stata fornita la definizione di IE secondo MUC, cioè come suddivisione in 5 sottoattività e, ad ognuna, è stata dedicata una breve analisi nella quale sono stati approfonditi i target di tale attività.
Il capitolo successivo è stato destinato interamente alla descrizione di GATE, uno strumento robusto per l'Information Extraction con particolare attenzione al suo modulo ANNIE, A Nearly New IE system, volto principalmente alla Named Entity recognition, una delle 5 sottoattività descritte precedentemente. Dopo un iniziale percorso attraverso gli elementi caratterizzanti il sistema GATE, e quindi una comprensione teorica dei suoi perché, ci siamo spostati verso ANNIE; di tale modulo abbiamo analizzato le Processing Resources, cioè quelle risorse che vanno a comporre la pipeline che effettivamente realizza la resolution delle entità (Tokeniser, Gazetteer, Sentence Splitter, Part of Speech Tagger, Semantic Tagger, OrthoMatcher e Pronominal Coreference): per ogni PR sono state date le conoscenze degli algoritmi usati e le descrizioni delle categorie di dati con le quali esse lavorano. Infine, l'ultima parte del capitolo è stata dedicata all'analisi degli strumenti che GATE mette a disposizione per effettuare una valutazione delle prestazioni, cioè l'AnnotationDiff Tool e il Benchmarking Tool.
Nel capitolo seguente tutte le conoscenze acquisite sono state messe in pratica utilizzando effettivamente GATE per la Named Entity recognition; come documenti sui quali lavorare ci siamo serviti di alcuni siti web facenti parte del progetto WISDOM di cui ha fatto parte il DBGROUP della Professoressa nonché relatrice Bergamaschi. I siti (in inglese) sono stati analizzati e processati dalle Processing Resources che hanno restituito l'output relativo alle query in questione. Per verificare e toccare con mano il lavoro di tali PR, abbiamo salvato i documenti in formato XML e li abbiamo esplorati: sono stati molti i frammenti d'interesse nei quali si è potuto vedere messe in pratica le tecniche imparate in modo teorico in precedenza; frammenti che sono stati riportati nel testo correlati da spiegazioni e richiami alla parte di teoria.
Terminata questa fase, si è proceduto con la valutazione delle prestazioni del sistema, utilizzando uno dei due strumenti messi a disposizione da GATE: l'AnnotationDiff Tool. Tale strumento implica due testi da confrontare, e quindi si è deciso di organizzare la valutazione confrontando il sito in inglese con il corrispondente in italiano, il sito in inglese referenziato (cioè annotato a mano) con il sito in inglese annotato dal sistema, il sito in italiano referenziato con il sito in italiano annotato dal sistema, nonostante il sistema non fosse settato per l'analisi di testi in italiano. Per ognuna di queste valutazioni la percentuale di resolution non ha mai superato il 58% che rappresenta una percentuale molto bassa e in molti casi inaccettabile.
In conclusione, anche se il sistema di default viene rilasciato già pronto per effettuare la resolution in lingua inglese, non può essere preso ed utilizzato così com'è (se non con queste prestazioni) ma ha bisogno di un raffinamento, di un'attenzione dedicata agli aspetti relativi all'ambito di impiego: se il mio sistema verrà utilizzato per la risoluzione dei nomi

di persona, dovrò curare in modo significativo la lista del Gazetteer che contiene tali nomi, modificando la versione di base fornita col sistema ed inserendo nuove entry; non solo, molto probabilmente dovrò anche affinare le regole grammaticali del POS e del Semantic Tagger, renderle più precise e specifiche al mio scopo. Tutte queste modifiche vanno poi testate con l'attività di training, che consiste nel provare il sistema con centinaia e centinaia di testi diversi nel tipo e nel registro: capite bene che prima di ottenere un sistema che performi al massimo dell'efficacia bisogna lavorarci parecchio. Lo stesso discorso vale per la resolution nelle altre lingue: negli ultimi anni lo sforzo è stato diretto verso la recognition multilingua, ma gli strumenti prodotti sono "grezzi" ed hanno bisogno di un profondo raffinamento.
Per quanto riguarda gli sviluppi a venire, una delle direzioni future in questo campo è l'integrazione delle processing resources che apprendono sullo sfondo, mentre l'utente sta annotando un corpus nell'ambiente grafico di GATE. Attualmente, i modelli statistici possono essere integrati ma necessitano di essere sviluppati separatamente. Si sta anche estendendo il sistema con moduli per la generazione della lingua, in modo tale da permettere la costruzione di applicazioni che richiedono la produzione della lingua in aggiunta all'analisi, es. la generazione di report intelligenti da dati IE.

INDICE DELLE FONTI

- **H. Cunningham, D. Maynard, V. Tablan, K. Bontcheva, C. Ursu, M. Dimitrov, M. Dowman, N. Aswani, I. Roberts : "Developing Language Processing Components with GATE Version 3 (a User Guide)"**, 2006

- **H. Cunningham : "GATE: an Architecture for Development of Robust HLT Applications"**, 2006

- **H. Cunningham : "Information Extraction, Automatic"**, 2006

- **D. Maynard : "Multilingual adaptation of ANNIE, a reusable information extraction tool"**, 2006

- **V. Tablan : "Named Entity Recognition from Diverse Text Types"**, 2005

- **B. M. Sundheim, N. A. Chinchor : "Survey of the Message Understanding Conferences"**, 2000

- **http://www.wikipedia.org**

- **http://www.gate.ac.uk**

- **http://www.booking.com**

- **http://www.venere.com**