TECNICHE DI ELABORAZIONE DEL LINGUAGGIO NATURALE (ELN) NEL TEXT MINING

di Michele Rubino

Corso di ELN, A. A. 2003/2004

Data Mining

- Il Data Mining, noto anche come Knowledge Discovery in Databases (KDD) è un aspetto ben sviluppato della Business Intelligence; le applicazioni di KDD scoprono i trend e i pattern in grandi collezioni di dati (Data Warehouses) in modo semiautomatico. Per l'analisi di testi al DM si preferisce la più recente disciplina del Text Mining, che analizza un testo non solo come sequenza di caratteri, ma come compagine di significati

Text Mining

- Il Text Mining è la ricerca di informazioni su raccolte di testo scritto in linguaggio naturale secondo schemi simili a quelli del Data Mining più altri specifici per il linguaggio naturale. Per quanto l'utilizzo più redditizio sia quello aziendale, per analizzare e-mail contenenti feedback dai clienti, documenti e presentazioni, articoli giornalistici e dati economici, le potenzialità del TM sono sfruttate in numerosi campi scientifici e per applicazioni di uso comune

Perché l'ELN è importante per il Text Mining

- Essendo i testi scritti in linguaggio naturale è necessario disporre di un sistema che analizzi quest'ultimo (e non si limiti a considerarlo come una semplice sequenza di caratteri
- L'elaborazione più semplice è quella dell'identificazione delle parole di una determinata lingua; generalizzando da parole a multiword, a frasi, a significato ed a contesto, un testo fornisce grandi quantità di informazioni non esplicite o esprime concetti uguali in forme diverse

Perché l'ELN è importante per il Text Mining

- Analizzando le parole e le multiword si possono effettuare studi sulla semantica di un testo (essenziale per i motori di ricerca)
- Comprendere il significato di una frase permette di ricavare dati dalla medesima in modo automatico
- Dal contesto della frase si possono ottenere dati più precisi e risolvere le ambiguità, che sono un elemento distintivo del linguaggio naturale rispetto ai linguaggi artificiali
- Dalla retorica di un articolo di politica interna è possibile capire l'allineamento politico dell'autore!

Tecniche di ELN usate nel Text Mining

- String matching con espressioni regolari
- Part of Speech Tagging per l'analisi grammaticale
- Parsing per l'analisi sintattica (tramite utilizzo di Grammatiche e del POS tagging medesimo)
- Word Frequency, per analisi statistiche e considerazioni sullo stile
- Latent Semantic Analysis, per ricerca e ranking di testi (tramite espressioni regolari e WF)

Tecniche di ELN usate nel Text Mining

- Reference Resolution per la comprensione del testo
- Rhetorical Structure Theory, per l'analisi dello stile dell'autore
- Information Integration, per la costruzione automatica di basi di dati partendo da documenti scritti in linguaggio naturale
- Active Summaries, per la costruzione automatica di basi di dati contenenti linguaggio naturale

Problemi delle tecniche di ELN

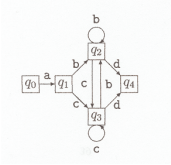
- Variazioni morfologiche: quasi tutti gli algoritmi di POS tagging e parsing lavorano sui lemmi delle parole da analizzare. Le variazioni morfologiche possono ingannarli
- Word Frequency: la discontinuità della frequenza delle parole mette in difficoltà gli algoritmi di POS tagging
- Collocazioni: individuare le collocazioni delle parole e' necessario per algoritmi avanzati di LSA

Espressioni regolari

- Le espressioni regolari sono una tecnica estremamente comune, essendo alla base dei motori di ricerca. Ad esempio l'espressione ".* Intel .*" restituisce tutti i documenti in cui si trova la parola Intel (essendo questo il caso più comune nei motori di ricerca si omettono gli asterischi)
- Una ricerca più raffinata potrebbe riguardare le schede di rete Intel… ".* Intel PRO [.*/M]|[wireless .*] " troverebbe documenti in cui sono contenute frasi come " Intel PRO 100/M" o "Intel PRO wireless 2011B"

Espressioni regolari

- Le espressioni regolari possono essere rappresentate come stringhe o come macchine a stati finiti. Per esempio il pattern "una A seguita da un numero qualsiasi di B e C e terminata da una D" può essere scritta come 'a[bc]*d' o con la macchina seguente



Espressioni regolari

- Una macchina a stati finiti consiste di un set di stati (i vari q_i) e di mosse verso altri stati quando si incontrano le lettere che etichettano le transizioni
- L'insieme di tutte le stringhe di testo che corrispondono a una particolare espressione regolare costruita da un alfabeto Σ è detto linguaggio regolare. La classe dei linguaggi regolari è descritta dai seguenti assiomi:

Assiomi per i linguaggi regolari

- L'insieme vuoto ø è un ling. regolare
- Per ogni elemento x dell'alfabeto {x} è un ling. regolare
- Se L1 e L2 sono ling. regolari allora $L1 \cdot L2=\{xy|x \in L1, y \in L2\}$ è un ling. regolare
- Se L1 e L2 sono ling. regolari allora $L1 \cup L2=\{x|x \in L1 \text{ or } x \in L2\}$ è un ling. regolare
- $L1^*=\{x^n|x \in L1, n \in \mathbb{N}\}$, la chiusura di Kleene di L1, è un ling. regolare

Espressioni regolari e annidamento

- Normalmente la strategia di matching è quella definita nello standard POSIX come “leftmost longest match”, cioè si testano tutti i possibili suffissi della stringa per il prefisso che corrisponde al pattern
- Tuttavia per i linguaggi di contrassegno in cui si possono scrivere record annidati (come l’XML) si utilizza la strategia “shortest non-nested”, cioè si segnala il match più piccolo possibile che non sia completamente incluso in un altro possibile match

Espressioni regolari e annidamento

```
<persona>
    <nome>Michele Rubino</nome>
    <padre>
        <nome>Nicola Rubino</nome>
    </padre>
</persona>
<persona>…</persona>
…
</persona>
```

- Fra questi record potremmo cercare con il leftmost longest match tutte le persone che si chiamano Michele scrivendo “<persona><nome>Michele .*</nome> .* </persona>

Espressioni regolari e annidamento

```
<persona>
    <nome>Michele Rubino</nome>
    <padre>
        <nome>Nicola Rubino</nome>
    </padre>
</persona>
<persona>…</persona>
…
</persona>
```

- Rosso = match dell’espressione; Verde = nome del match
- Il risultato non è soddisfacente… non solo il match del nome è troppo lungo e ha preso due nomi a livelli diversi, ma in totale la query restituisce un solo match (anche se ci fossero stati altri Michele nel database!). Per questo dobbiamo usare il paradigma “shortest non-nested”

Word Frequency

- La frequenza delle parole è una delle cose più semplici da studiare
- Può essere utilizzata per statistiche sulla ricchezza del vocabolario dell’autore o per indici di leggibilità
- La WF è fondamentale per il funzionamento degli algoritmi di massima verosimiglianza nel POS Tagging
- La WF è importante per algoritmi avanzati di Latent Semantic Analysis
- La WF è anche un problema di grande interesse teorico per lo studio di algoritmi di matching e sorting (i risultati di una “competizione fra linguaggi” per il calcolo della word frequency si trovano a http://www.bagley.org/~doug/shootout/bench/wordfreq/)

Word Frequency

- Uno dei problemi più importanti per gli algoritmi di massima verosimiglianza è la "discontinuità" della frequenza delle parole: le parole con frequenza bassa (in Tom Sawyer il 50% delle parole compare una volta sola e il 90% compare meno di 10 volte) li mettono in difficoltà. Per questo è necessario effettuare uno "smoothing" delle probabilità.
- Con l'algoritmo di Good-Turing si considera il numero di parole (Nc) che hanno frequenza c. La frequenza "smoothed" corrispondente è c*= (c+1) [N(c+1)/Nc]

Variazioni morfologiche

- Un altro problema per gli algoritmi di POS tagging è data dalle variazioni morfologiche
- Esistono tre tipi di variazioni morfologiche:
- Inflessionali: una parola è derivata dal lemma, e acquisisce caratteristiche addizionali ma mantiene la sua funzione (es. dog $\rightarrow$dogs)
- Derivazionali: la funzione della parola è cambiata (es. able $\rightarrow$ably)
- Compositive: parole indipendenti sono unite per formarne una nuova (es. button+hole $\rightarrow$buttonhole)

Variazioni morfologiche

- Per eliminare le variazioni morfologiche e ridurre le parole al solo lemma esistono numerosi algoritmi di analisi morfologica, da utilizzare sul testo prima del tagging

Part of Speech Tagging

- Ogni parola ha una funzione grammaticale detta "parte del discorso", contrassegnata con un tag; sfortunatamente una parola può corrispondere a più parti del discorso (es. will è sia un sostantivo che un verbo). Per disambiguare si fa ricorso a varie tecniche:
- 1. Massima verosimiglianza: analizzando il corpus di addestramento si annota il tipo (la parte del discorso) più frequente per ogni parola; questo tipo viene applicato a tutte le future istanze della parola trovate dal tagger. Se la parola non è nel corpus la si considera "nome proprio"

Part of Speech Tagging

- 2. Hidden Markov Taggers: Oltre ai dati raccolti per la massima verosimiglianza, si raccolgono statistiche sulla probabilità che un tag ne segua un altro. Per esempio, “can” è solitamente un verbo, ma il tag “verbo” non è mai preceduto da un tag “articolo”, perciò la coppia “articolo+sostantivo” (can sarebbe il sostantivo) è molto più probabile. Con queste considerazioni il tagging è corretto nel 95% dei casi (il punteggio ottenuto da tagger umani è 98%)

Part of Speech Tagging

- 3. Transformational Taggers: si scrive a mano una serie di regole che correggono errori di tagging (per esempio: “cambia in sostantivi i Verbi che si trovano dopo gli articoli”) e si applicano le regole di correzione sull’output di un altro tagger (per esempio uno basato sulla massima verosimiglianza)

Grammatiche

- Una grammatica è un modo conciso di specificare le relazioni fra frasi che sono accettabili in un linguaggio. Solitamente consistono di un insieme di simboli terminali (parole del linguaggio), un insieme di simboli non terminali (frasi del linguaggio), un simbolo non terminale iniziale e un insieme di regole che definisce le relazioni fra i simboli terminali (indicati con T) e quelli non terminali (indicati con NT). Chomsky ha ordinato le grammatiche in quattro categorie con crescente “potere generativo” (espressività).

Grammatiche

- 1. (Left) regular grammars: equivalenti agli automi a stati finiti per espressività, hanno solo regole del tipo $NT \rightarrow T\ NT_1 \ldots NT_n$
- 2. Context free grammars: non richiedono che il primo simbolo sia un terminale, le regole sono del tipo $NT \rightarrow NT_1 \ldots NT_n$
- 3. Context sensitive grammars: hanno più simboli nella parte sinistra della regola $NT_a \ldots NT_z \rightarrow NT_1 \ldots NT_n$ dove $n>=z$
- 4. Recursively enumerable grammars: non hanno la restrizione su n e z. Equivalgono per espressività alle macchine di Turing

Parsing

- Il parsing è il processo di connessione fra i tag in una struttura ad albero: i nodi rappresentano i sintagmi, gli archi rappresentano l'applicazione delle regole grammaticali e le foglie rappresentano le parole. A differenza di quanto avviene nei linguaggi di programmazione, una frase in linguaggio naturale può avere più di un albero di parsing valido. Questo richiede un processo di disambiguazione…

Parsing

- 1. Probabilistic Context-Free Grammars: la grammatica ha una probabilità associata ad ogni regola (calcolata in fase di addestramento); la disambiguazione si ottiene scegliendo l'albero di parsing che massimizza la produttoria delle probabilità associate alle regole scelte.

Parsing

- 2. Markov grammars: anziché calcolare la probabilità per ogni regola si calcola $p(c_i|c_{i-1})$ che il componente c_i (T o NT) compaia nell'espansione del non-terminale n, tenendo in considerazione il precedente componente dell'espansione, c_{i-1}. Per scegliere la regola più probabile possiamo imporre regole del tipo p(pronome|SV,Verbo) < p(SN|SV,Verbo); infine l'albero di parsing può essere selezionato con regole "à la PCFG": $p(\text{regola}|n) = \prod p(c_i|n, c_{i-1})$

Parsing

- 3. Lexicalized parsing: anziché effettuare il parsing sui tag, possiamo lavorare direttamente sulle parole del vocabolario secondo uno stile simile alle PCFG; per ridurre la complessità consideriamo solo la "testa" della frase: l'elemento nominale per i sintagmi nominali, la preposizione per i sintagmi proposizionali e così via

Parsing

- 4. Transformation based parsing: come nei transformational taggers: si effettua un parsing grossolano e si applicano regole scritte a mano per correggere gli errori

Latent Semantic Analysis

- La LSA si basa sull'identificazione dell'argomento di un testo in base alla presenza di parole chiave comuni ad altri testi dello stesso argomento; per esempio, trovare "human" "computer" "interface" "system" identifica un testo sulle HCI. La LSA non fa analisi sintattica, morfologica o di significato, ed è usata soprattutto per cercare sinonimi (parole che si trovano nello stesso contesto del lemma, ci sono tecniche per distinguerli dai contrari), per ranking di testi per argomento e per calcolare la correlazione fra documenti

Latent Semantic Analysis

- Per un ranking migliore e per effettuare ricerche più estese, la LSA sfrutta analisi della Word Frequency (soprattutto per il ranking) e delle collocazioni delle parole

Collocazione

- La collocazione è un uso di una frase che suggerisce al lettore un concetto non indicato dalle parole che la compongono: *a* "*disk drive" is disk shaped but does not drive anywhere*.
- La collocazione delle parole permette alla Latent Semantic Analysis di distinguere fra un testo sui computer e uno sulle auto cercando le stesse parole (disk, windows, engine, graphic): infatti il disco può essere un disco rigido (disk drive) o un freno a disco (disk brake), "windows" è un sistema operativo o un finestrino, il motore può essere grafico o a quattro cilindri e la parola graphic può essere un sostantivo (grafica) o un aggettivo (graphic meters, cioè indicatori grafici di velocità, benzina ecc.)

Collocazione

- L'analisi della collocazione è utilizzata anche per algoritmi di Information Integration (che vedremo in seguito)
- Più che cercare le collocazioni automaticamente è preferibile tenere un almanacco di multiword con le collocazioni annotate da usare durante la LSA

Reference Resolution

- Con la reference resolution si individuano nel contesto i riferimenti dei pronomi e dei termini quantificati. Per applicare algoritmi di RR è necessario trasporre un testo in QLF. Questa è una forma equivalente alla forma logica, perciò per ottenerla il primo passo è trasformare il testo in forma logica

Forma logica e QLF

- Definite Clause Grammars: nelle DCG le produzioni grammaticali del tipo $F \rightarrow SN\ SV$ sono trasformate in forma logica: ad esempio $F \rightarrow SN\ SV$ diventa $SN(s1) \wedge SV(s2) \Rightarrow F(append(s1,s2))$ ovvero dato un sintagma nominale s1 e uno Verbale s2 la loro concatenazione è una frase

Forma logica e QLF

- I simboli non terminali possono essere arricchiti con informazioni semantiche, scrivendo SN(sem) in una regola della DCG. Ora definiremo una grammatica per un sottoinsieme della lingua inglese

Forma logica e QLF

1. F(rel(obj)) →SN(obj) SV(rel)
2. SV(rel(obj)) →Verb(rel) SN(obj)
3. SN(ref(he)) →**he** | **him**
4. SN([∃ a a]) →**someone**
5. SN([∀a a]) →**everyone**
6. SN(Tom) →**Tom**
7. SN(Sam) →**Sam**
8. Verbo(λx λy Hears(x,y)) →**hears**
9. Verbo(λx Sneeze(x)) →**sneeze**
10. Verbo(ellipsis) →**did**

Forma logica e QLF

- Come si vede, hears ha semantica λx λy Hears(x,y). La regola 2 descrive la sintassi di un sintagma verbale come combinazione di un verbo e di un sintagma nominale. Inoltre afferma che la semantica del sintagma "hears Sam" è ottenuto applicando la semantica del verbo "hears" alla semantica del sintagma nominale "Sam". Ripetiamo il processo per la regola 1 e applichiamo la semantica del sintagma nominale "Tom" come argomento di "hears Sam" generando hears(Tom,Sam)

Forma logica e QLF

- Ci sono casi in cui la notazione abituale non è consigliabile; ad esempio nel caso di "Everyone hears Sam", finiremmo per generare SN(∀a SV(Hears(a,Sam))). Per evitarlo aggiungiamo nella grammatica regole con termini quantificati (come le regole 4 e 5) in modo da scrivere Hears([∀a a],Sam). Questa forma è chiamata Forma Quasi Logica, o QLF

Forma logica e QLF

- Per convertire da forma logica a QLF basta applicare due regole ai termini quantificati:
- 1. QLF(Univ) → ∀xP(x) ⇒Q(x)
- 2. QLF(Exist) →∃ xP(x) ∧Q(x)
- Anche qui compaiono delle ambiguità: "everyone hears someone" ha due forme logiche diverse a seconda che tutti sentano una persona in particolare o per conto proprio. Per disambiguare bisogna cercare il "qualcuno" nel contesto tramite la RR

Reference Resolution

- Con la reference resolution si individuano nel contesto i riferimenti dei pronomi e dei termini quantificati. La transizione dalla QLF alla Resolved Logic Form (RLF) è rappresentata da equivalenze condizionali

Reference Resolution

- Equivalenze condizionali:

QLF ⇔RLF
if condition 1
…
condition n

Cioè le due forme corrispondono se e solo se le condizioni sono soddisfatte. Le equivalenze sono scritte a mano.

Un esempio di Reference Resolution

- Prendiamo un frammento di testo:

Frasi	Forme quasi logiche
Tom sneezes.	sneeze(tom)
Sam hears him.	hears(sam,ref(he))
Everyone did.	ellipsis([∀a a])

Un esempio di Reference Resolution

- Una semplice equivalenza condizionale per la RR dei pronomi è

Verbo(ref(Pronome)) ⇔ Verbo(Qualcuno)
if
contesto(C)
AltroVerbo(Qualcuno) = C
Pronome(Qualcuno)

Un esempio di Reference Resolution

- Abbiamo un match per la QLF della seconda frase: Verbo = λx hears(Sam,x) e Pronome = he
- La prima condizione è verificata da C = sneeze(Tom)
- La seconda estrae il soggetto con le corrispondenze AltroVerbo = sneeze e Qualcuno = Tom
- La terza è verificata da he(Tom) dato che Tom è effettivamente un "lui" (he). La corrispondente RLF è (λx hears(Sam,x)) (Tom), che si riduce a hears(Sam,Tom)

Un esempio di Reference Resolution

- Per risolvere l'ellissi della terza frase abbiamo bisogno di un'altra equivalenza:

ellipsis(X) $\Leftrightarrow$ Verbo(X)

if

contesto(C)

Verbo(Soggetto) = C

parallel(X, Qualcuno)

Un esempio di Reference Resolution

- Nel nostro esempio il matching restituisce X = [$\forall$a a] e C = hears (Sam,Tom) dato che abbiamo completato l'analisi dei precedenti periodi
- La seconda condizione è soddisfatta da Verb = λx hears(Sam,x) e Soggetto = Sam
- Il predicato parallel è verificato se i suoi argomenti possono essere usati in posti simili. Perciò se assumiamo che tutti possano fare ciò che fa Sam la forma logica della frase diventa $\forall$a hears(a,Tom) dopo la risoluzione del termine quantificato

Rhetorical Structure Theory

- Nella RST il computer funge da revisore di un testo scritto (una terza parte fra autore e lettore) e fornisce motivazioni sul perché gli elementi di un testo vi sono stati inclusi dall'autore. La base della RST è data dalla frammentazione del testo in span (proposizioni o periodi). Quando una span contiene un enunciato e un'altra span fornisce spiegazioni sull'enunciato la prima è detta nucleo e la seconda satellite (a indicare la maggiore importanza dell'enunciato); una relazione tra span di pari importanza è detta multinucleare

Rhetorical Structure Theory

- Le relazioni possono essere di:
- 1. Background (il satellite facilita la comprensione del nucleo)
- 2. Elaborazione (il satellite fornisce informazioni addizionali)
- 3. Preparazione (il satellite prepara il lettore ad aspettarsi il testo presentato)
- 4. Multinucleare
- 5. Contrasto fra due alternative

Rhetorical Structure Theory

- Per esempio la relazione di background indica che il revisore pensa che l'autore pensi che il lettore potrebbe non capire il nucleo ma che comprendendo il satellite (più facile da capire) arriverà a un grado soddisfacente di comprensione del nucleo

Information Integration

- La Information Integration consiste nella raccolta di dati strutturati (cXML, xCBL) e non strutturati riguardanti prodotti o categorie di prodotti e nella classificazione dei dati raccolti in un Content Management System, secondo schemi standard (UN/SPSC) o decisi dal gestore del mercato che richiede il sistema di Information Integration (e del CMS)
- Il cliente deve avere la possibilità di definire viste personalizzate dei dati

Information Integration

- Il passo iniziale di Information Extraction si può fare manualmente, in modo semiautomatico (si da al sistema uno schema dei layout usati dalle aziende in modo che sappia dove cercare e si analizzano le collocazioni dei dati) o in modo automatico. In questo caso sono spesso utilizzati algoritmi di reti neurali che cerchino dati scritti in forme simili a quelle viste in fase di addestramento
- La classificazione delle informazioni secondo lo schema previsto può essere fatta con regole definite dall'utente o con reti neurali

Active Summaries

- I sistemi di Business Intelligence spesso riorganizzano i database inserendovi metadati che sono utilizzati per generare indici multidimensionali. Una data warehouse organizzata in questo modo è detta OLAP cube (OLAP = On Line Analytical Processing)

Active Summaries

- L'idea di un Active Summary è quella di replicare l'infrastruttura OLAP cube per una collezione di documenti di testo. Questo implica l'identificazione delle relazioni semantiche all'interno del documento e fra documenti e la generazione di viste multidimensionali per facilitare la produzione di rapporti e abbassare il tempo di risposta delle query su un database di testi

Esempio: analisi automatica di rapporti commerciali

- Per fare un esempio dell'uso pratico delle tecniche di elaborazione del linguaggio naturale nel Text Mining, prendiamo un utilizzo comune soprattutto nel settore degli uffici: l'analisi automatica di rapporti e recensioni per confrontare materiali da acquistare

Esempio: analisi automatica di rapporti commerciali

- Supponiamo ad esempio che l'utente sia interessato ad acquistare dei processori per computer e possieda una base di dati con le caratteristiche di un certo numero di processori esistenti. Avendo notizia dell'arrivo sul mercato dei nuovi modelli delle due principali case produttrici, l'Inter Pintimonium e l'AMC Atletico, desidera confrontarli con quelli che ha già valutato.

Esempio: analisi automatica di rapporti commerciali

- Il primo passo di questa ricerca consiste nella ricerca di recensioni di processori che descrivano le prestazioni e i dati tecnici. Per fare questo si affida ad un motore di ricerca basato su Latent Semantic Analysis, addestrato ad identificare un documento del genere cercando le parole “processore Inter AMC prestazioni dati” e facendo opportunamente il ranking per poi analizzare più accuratamente i documenti di maggiore rilevanza.

Esempio: analisi automatica di rapporti commerciali

- Una volta trovato il documento più rilevante, il sistema può tentare di estrarre automaticamente i dati tecnici e metterli nel database dell’utente tramite l’Information Integration, secondo schemi già definiti dall’utente. Per i processori esistono due schemi:
- Lo schema 1 è dato da: Casa produttrice, Nome, Connessione alla scheda madre, Prezzo, Disponibilità
- Lo schema 2 ha gli stessi dati più la frequenza del Clock
- Il sistema è stato addestrato a riconoscere i dati tecnici scritti in varie forme con algoritmi di reti neurali

Esempio: analisi automatica di rapporti commerciali

- Il sistema trova due tabelle in formato di testo semplice, ma corrispondente a un pattern visto in addestramento:

```
Inter Pintimonium
Casa produttrice: Inter
Nome: Pintimonium
Connessione alla scheda madre: Socket 378
Prezzo: 300 €
Disponibilità: già disponibile

AMC Atletico
Casa produttrice: AMC
Nome: Atletico
Connessione alla scheda madre: Socket A
Prezzo: 400 €
Disponibilità: in uscita a Novembre
Clock: 2,4 Gigahertz
```

Esempio: analisi automatica di rapporti commerciali

- I due processori sono riconosciuti rispettivamente con lo schema 1 (il Pintimonium) e con lo schema 2 (l’Atletico)
- Leggendo i dati, l’utente pensa di cercare altre informazioni sul Pintimonium, che sembra un buon candidato. È interessato in particolare alla frequenza del clock del processore Inter.
- Per questo compito è necessario cercare nel testo. Una volta trovata la parola “clock”, il sistema prova ad analizzare la frase che la contiene per stabilire a quale dei due processori si riferisce.

Esempio: analisi automatica di rapporti commerciali

- La frase in questione è la seguente: `Il Pintimonium possiede un clock a 3 Gigahertz`
- Per capire chi è che ha il clock suddetto, è necessaria la Reference Resolution
- Dal parsing della frase e dalla conseguente trasformazione in forma logica si vede che il clock è a 3 Gigahertz e che questo clock è una caratteristica del Pintimonium
- Poiché la frase è di interesse (parla di Pintimonium e di un clock ad esso relativo che gli fa da complemento oggetto), si estrae il sintagma nominale del complemento oggetto (con annesso sintagma proposizionale sulla frequenza), cioè "clock a 3 Gigahertz", e lo si sottopone all'algoritmo di Information Integration perché riconosca il pattern con cui si indica la frequenza del clock.

Esempio: analisi automatica di rapporti commerciali

- Sebbene i dati sui processori siano finalmente completi, l'analisi può andare ancora avanti: con la Rhetorical Structure Theory è possibile rilevare che il testo è composto in larga parte di relazioni multinucleari, salvo che in due zone in cui l'autore cerca di giustificare le sue osservazioni (molte relazioni di background e preparazione)
- Il sistema può dunque indicare all'utente i due stralci di testo, che potrebbero contenere elementi importanti.

Esempio: analisi automatica di rapporti commerciali

- Gli stralci sono:

`Il Pintimonium possiede un clock a 3 Gigahertz. Grazie a questa impressionante frequenza di clock e alle innovative reti logiche dedicate alla moltiplicazione, il processore nerazzurro si è comportato meglio dell'Atletico sul benchmark riguardante i calcoli in virgola mobile (particolarmente importanti per la grafica 3D, e quindi per i giochi).`

e

`L'Atletico tuttavia dispone di un asso nella manica: il processore appenninico incorpora infatti una cache di primo livello Thornetown di ben 256 Kilobytes. Questa permette al processore di leggere e scrivere i dati a una velocità 20-30 volte superiore a quella dettata dal normale collegamento processore-memoria principale; di conseguenza le prestazioni complessive dell'Atletico si sono dimostrate superiori in quasi tutti gli altri test`

Esempio: analisi automatica di rapporti commerciali

- Un paragrafo scartato perché retoricamente meno intenso è proprio quello introduttivo:

`Ancora una volta le due principali case produttrici di processori, l'Inter (che spera di riprendersi dai recenti risultati calcistici, alquanto deludenti, con un successo commerciale nel campo dei processori) e l'AMC (l'Azienda per la Mobilità Catanzarese, che oltre a produrre chip gestisce i trasporti autoferrotramviari nel capoluogo calabrese), tornano a scontrarsi con le offerte autunnali di processori. Esigenze di mercato hanno imposto di anticipare la presentazione dei loro più recenti modelli già alla fine di Settembre (facendoci sperare di uscire con qualche nuovo processore per Natale).`

Esempio: analisi automatica di rapporti commerciali

- Con queste informazioni aggiuntive a sua disposizione, l'utente può effettuare una scelta più consapevole fra i due modelli di processore, con un notevole risparmio di tempo (e di denaro, per le aziende) e con notevoli facilitazioni nel confronto fra processori papabili per l'acquisto

Bibliografia

- S. Williams, A Survey of Natural Language Processing Techniques for Text Data Mining
- M. Bleyberg, Inference Rules for Text Data Mining
- S. Harabagiu, Text and Knowledge Mining for Coreference Resolution
- D. Lin, Identifying Synonyms Among Distributionally Similar Words
- D. Lin, Discovery of Inference Rules for Question Answering
- D. Lin, Induction of Semantic Classes from Natural Language Text
- D. Lin, DIRT – Discovering of Inference Rules from Text
- D. Lin, Concept Discovery from Text
- D. Fensel et al. , A Knowledge Level Analysis of Information Integration Issues in B2B Electronic Commerce