Brill Tagger:

Implementazione di un algoritmo di Machine Learning per il PoS Tagging

Anno Accademico 2007/2008
Corso di Elaborazione del Linguaggio Naturale
Prof. Amedeo Cappelli

PARTIAMO DALL'INIZIO …

Qualche delucidazione

Part of Speech (PoS)

- **Categoria morfo-sintattica cui una parola appartiene**
  - Le categorie principali sono:
    - Nomi, verbi, aggettivi, avverbi, articoli, pronomi, congiunzioni, …
- **Il PoS Tagging è:**
  - Processo di assegnazione della Part of Speech ad ogni parola di un corpus (insieme di documenti)
- **E' utile a varie applicazioni del NLP:**
  - come passo precedente al parsing sintattico;
  - in information extraction and retrieval;
  - text to speech systems;
  - etc.

Relazioni con la morfologia

- L'analisi morfologica consiste, data una parola, nel trovare tutte le sue interpretazioni morfologiche.
- Un analizzatore morfologico però non sa decidere quale analisi è giusta e quale è sbagliata, perciò dà tutte le analisi probabili di quella parola.
- E' quindi presente ambiguità:
  - Es. porta N FS
  - porta V 3PS

Relazioni con la Morfologia (2)

Un tagger che effettui il POS Tagging, data una parola, trova la sua unica interpretazione morfologica: il suo lavoro consiste in

Analisi morfologica + Disambiguazione

Ambiguità e Disambiguazione

- L' ambiguità è un fenomeno pervasivo del linguaggio
- La lingua italiana, fortemente flessiva, possiede moltissimi casi di parole omografe, cioè che presentano la stessa grafia e hanno significati o funzioni diverse.

Ambiguità e Disambiguazione(2)

- **Ambiguità grammaticale**: appartiene alle forme che esplicano nella frase la stessa funzione, hanno lo stesso significato ma individuano modo, persona, genere o numero diversi
  - Sostantivi: "attaccapanni"= sing | plur "portaborse"=sing| plur | masch| femm
  - Verbi: "venga"=1ª| 2ª| 3ªpers,cong.pres.
- **Ambiguità funzionale**: forme che esplicano nella frase funzioni diverse
  - "fatto"= verbo|aggettivo|sostantivo
- **Ambiguità lessicale**: relativa ad omografie di tipo semantico
  - Omografi non omofoni: "bòtte"|"bótte"
  - Parole polisemiche: "mobile" (il mobile | la squadra mobile), "fronte"
- **Ambiguità multiple**: sovrapposizione di ambiguità sopra descritte
  - "faccia"=
    - Lessicale (viso | superficie di un poliedro)
    - Funzionale(sostantivo | verbo)
    - Grammaticale(1ª|2ª| 3ªpers,cong.pres.)

Ambiguità e Disambiguazione (3)

Per gli esseri umani la disambiguazione è parte fondamentale della competenza linguistica, che consiste nel saper riconoscere la categoria appropriata delle espressioni linguistiche nei vari contesti

Che cosa deve sapere un tagger per avere la stessa competenza?

Che cosa deve sapere un tagger per avere la stessa competenza?

- Quello che noi sappiamo (implicitamente) è che:
  - Non tutti i tag di una parola sono equiprobabili: alcuni tipi di analisi sono più frequenti di altre.
    - **Mondo**
    - **à MONDO#S@MS# (Sost,Masch. Sing)**
      à MONDO#A@MS# (Agg,Masch. Sing.)
      à MONDARE#V@S1IP# (Verbo,1pers.sing,Ind.Pres.)
  - non tutte le sequenze di tag sono legittime
    - **vincoli sintagmatici locali**
      - **Art + (Agg)*+ Sost OK!**
      - **Art + V NO!**
      - **V+ Art+ Avv NO!**
      - **Pron + V OK!**

Che cosa deve sapere un tagger per avere la stessa competenza?

- Quello che un tagger deve imparare è:
  - Ogni parola deve avere una sola PoS assegnata
  - Se esistono più PoS allora:
    - deve *disambiguare*, restituendo se possibile un solo tag:
      - »Utilizzando evidenze contestuali
      - »Utilizzando evidenze probabilistiche da corpora annotati

Algoritmi di Tagging

1. **Algoritmi a regole** regole di disambiguazione sviluppate manualmente da linguisti e messe in un grande database: si parte da poche centinaia a molte migliaia di regole. Ciò comporta anni di lavoro.

   TAGGIT (1971)
   ENGCG (1995)

2. **Algoritmi stocastici** la PoS corretta viene determinata a partire da evidenza statistica estratta da un corpus: consiste nello scegliere la sequenza di tag che ha la più alta probabilità rispetto ad altre sequenze. I taggers di questo tipo sono detti anche Hidden Markov Model e sono i più utilizzati per fare PoS tagging perché hanno un'accuratezza molto alta.

   CLAWS (1987)

Algoritmi di Tagging(2)

3. **Algoritmi di Machine Learning** le regole di disambiguazione sono indotte automaticamente da un corpus di training annotato. In particolare vengono apprese regole di trasformazione che riducono al massimo gli errori commessi inizialmente con l'applicazione del tag più probabile assegnato statisticamente.

BRILL TAGGER (1992)

Il lavoro di un tagger con algoritmo ML

- **Un tagger che implementi un algoritmo di Machine Learning impara a classificare correttamente le espressioni linguistiche attraverso un'opera di generalizzazione induttiva da un insieme di dati**
- Componenti di un algoritmo di machine learning**:**
  1. insieme di dati linguistici di partenza (training corpus)
  2. metodologia per indurre dai dati linguistici un modello generale dell'organizzazione e della struttura linguistica
  3. nuovi dati a cui si applica il modello appreso (test corpus)

Il lavoro di un tagger con algoritmo ML(2)

- **Metodologie di apprendimento**
  - unsupervised learning
    - **i dati linguistici di partenza non sono annotati linguisticamente**
  - supervised learning
    - **i dati linguistici di partenza sono annotati linguisticamente**

Brill Tagger

- **Inventato da Eric Brill nel 1992**
- **Tagger a regole apprese automaticamente con un algoritmo di machine learning**
  - Transformation Based Learning (TBL)
  - apprendimento supervised
- "*In recent years there has been an explosion of research in machine learning on finding ways to improve the accuracy of supervised classifier learning methods. An important finding is that a set of classifiers whose individual decisions are combined in some way (an ensemble) can be more accurate than any of its component classifiers if the errors of the individual classifiers are sufficiently uncorrelated.*"

["Improving Accuracy in Word Class Tagging through the Combination of Machine Learning Systems" Hans van Halteren, Jakub Zavrel, Walter Daelemans**]**

TBL

- Transformation-based error-driven learning (Brill 1993a, 1995), più comunemente chiamato "transformation-based learning" o TBL, è una tecnica basata sul Machine Learning.
- L' output di un TBL è una lista ordinata di regole che serve a ridurre gli errori.
- *"The best-known application of TBL has been to the task of part-of-speech tagging (Brill 1992, 1994)."*
- *"TBL has also been applied to a number of diverse linguistic tasks such as resolving syntactic attachment ambiguities (Brill and Resnik 1994), syntactic parsing (Brill 1993b), and word sense disambiguation (Dini et al 1998)."*

["Combining decision trees and transformation-based learning to correct transferred linguistic representations" Simon Corston-Oliver and Michael Gamon**]**

Apprendimento Supervised

- Supervised learning: approccio molto comune in machine learning e linguistica computazionale
- Non limitato al solo PoS tagging, anche se quest'ultimo ne è esempio classico.
- Invece di formulare esplicitamente regole per tagging:
  - annota a mano un po' di dati (gold standard);
  - lascia che il tagger estragga generalizzazioni da dati annotati;
  - usa il tagger "addestrato" su dati annotati per annotare nuovi dati

Brill Tagger-Training

1.Prima annotazione del testo. Per ogni parola:

A. Se è presente nel lessico, scegli il tag migliore utilizzando la strategia *Most Frequent Tag:*

IDEA: Le parole ambigue utilizzano un tag più spesso di altri.

- **Metodo:**

I. Creare un dizionario(un tagset) e annotare manualmente un corpus

II. Per ogni parola ambigua in un nuovo testo calcolare probabilità di annotazione secondo la formula:

I. $$P(tag \mid w) = \frac{|w/tag|}{Fw}$$

III. Assegnare il tag più probabile

Brill Tagger-Training

B. Altrimenti utilizza regole lessicali *(lexical rules*) per derivare il possibile tag

- Regole lessicali:
  - Utilizzate quando una parola non è nel lessico
  - Cercano di *indovinare* la classe della parola in base a informazioni morfologiche
  - solitamente viene assegnato un tag <nomeComune>
  - se la parola inizia per maiuscola, il tag è <nomeProprio>

Brill Tagger-Training

- Altre regole lessicali per individuare una parola sconosciuta:
  - trasformazioni di tag basandosi sulle lettere dei prefissi e dei suffissi:
    - Cambia il tag *a* della parola sconosciuta *W* con il tag *b* quando:
      - I primi (ultimi) 1,2,3,4 caratteri sono la lettera *z*
      - Togliendo (aggiungendo) il suffisso (prefisso) si ottiene una parola nota
      - Etc..
    - Questo metodo riesce a taggare correttamente:
      - 97.7 % parole note
      - 82.2 % parole sconosciute
      - 96 % totale parole

Brill Tagger-Training

2. Applicazione al testo delle *regole contestuali.* Per ogni parola:
- Viene cambiato il tag assegnatogli inizialmente in base alle parole circostanti
- In particolare vengono applicati gli schemi di trasformazione (templates), instanziandoli a turno su tutti i tag presenti nel Tagset
- Esempio di templates:

Trasforma	In	Se
X	Y	PREVTAG Z
X	Y	PREV1OR2TAG Z
X	Y	SURROUNDTAG Z Q
X	Y	PREVBIGRAM Z Q
…		

Brill Tagger-Training

- ***Esempi di abbreviazioni utilizzate nei templates:***

1. PREV – la parola precedente
2. PREVTAG ---la parole precedente è taggata X
3. PREV1OR2TAG ---una delle due parole precedenti è taggata X
4. PREV1OR2OR3TAG ---una delle tre parole precedenti è taggata X
5. WDAND2AFT ---la parola corrente è A e le due successive è B
6. PREV1OR2WD ---una delle due parole precedenti è A
7. NEXT1OR2TAG ---una delle due parole successive è taggata X
8. NEXTTAG ---la parola successiva è taggata X
9. NEXTWD ---la parola successiva è A
10. WDNEXTTAG ---la parola corrente è A e la successiva è taggata X
11. SURROUNDTAG ---la parola precedente è taggata X e la parola successiva è taggata Y
12. PREVBIGRAM ---le due parole precedenti sono taggate X
13. CURWD ---la parola corrente è A

Brill Tagger-Training

- E' importante notare che non c'è bisogno di essere accurati quando si costruisce la lista dei templates. Aggiungere un pessimo template non peggiorerà la performance del tagger.
- Infatti, se un template non è corretto, semplicemente nessuna delle sue regole verrà memorizzata come regola appresa dal tagger.
- Ciò rende più facile sperimentare più di un templates sul tagger.

Brill Tagger-Training

3.**Come vengono apprese le regole contestuali ?**

- Abbiamo un Corpus annotato manualmente (*gold standard*) che rappresenta l'output "corretto" di riferimento
  - La trasformazione scelta è quella che migliora al massimo l'accuratezza (Accuracy) rispetto al gold standard

$$Accuracy = \frac{output\_corretti}{|test\_corpus|}$$

  - **Esempio di valutazione**
    - |test_corpus| = 1200 tokens
    - output corretti del sistema = 945
    - Accuracy = 945 / 1200 = 0,79 (79%)

Brill Tagger-Learning

Il Brill ha un algoritmo che gli permette di imparare dai suoi errori correggendosi pian piano.

L'apprendimento finisce quando il numero degli errori non diminuisce più.

- Per ogni coppia di tag prova ogni trasformazione *t*
- Quantifica gli errori
- Scegli (come nuova regola) la trasformazione che riduce di più l'errore

Brill Tagger-Learning

Input dell'algoritmo di apprendimento

L = lessico, ossia una lista di parole a cui vengono associati tutti i possibili tag di cui il più probabile è messo in evidenza rispetto agli altri

CA = corpus di training annotato

T = lista ordinata di trasformazioni (inizialmente vuota)

C = corpus di training senza i tag

S = schemi di trasformazione

Brill Tagger-Learning

Ciclo dell'algoritmo di apprendimento

1. **Il corpus *C* viene annotato usando il lessico *L***
2. **Vengono applicate le trasformazioni *T***
3. **Il risultato è confrontato con il *Corpus Annotato* e viene compilata una lista di errori di tagging in una "tabella di confusione"**.
   3.1 La tabella consiste di una tripla < *taga, tagb, number* >, che *indica il numero delle volte* che il tagger sbaglia a taggare una parola con tag_a *quando avrebbe dovuto taggarla come* tag_b.
4. **Per ogni errore si creano schemi di trasformazione *S* ottenendo *n* trasformazioni contestuali potenziali *t***
   4.1. ciascuna trasformazione potenziale *t* viene applicata a *C*, e viene calcolata la riduzione del numero di errori di tagging che si ottiene
   4.2. la trasformazione *t* che determina la riduzione maggiore di errore viene selezionata e aggiunta a *T*.
   4.3. si torna al punto (2)
5. **Il training termina quando non è possibile trovare trasformazioni che riducono gli errori di tagging sopra una soglia *k* (threshold)**

Ora siamo pronti per testare il tagger ma...

- **ATTENZIONE!** Il Tagger non va valutato sugli stessi dati che sono stati usati per l'addestramento!
- Conta la capacità di generalizzazione di un tagger, dunque esso va valutato su dati che non ha visto in fase di addestramento.
- Quindi il test corpus su cui applicare tutte le regole apprese sarà:
  - Un nuovo corpus
  - Parte del corpus di training non ancora annotato

Ora siamo pronti per testare il tagger ma...

- Potenziali pericoli di valutazione su dati già visti:
  1. Sopravvalutazione di performance (es.: un tagger magari riconosce tutti i nomi propri del training set perché li ha visti durante il training, ma non riconoscerebbe tutti i nomi propri in un altro corpus).
  2. Si favoriscono taggers che tendono all'overtraining (es.: se nel training corpus la parola *torta* è sempre seguita dalla parola *Sacher*, un tagger che predice che la parola torta è sempre seguita da nome proprio funzionerà molto bene con lo stesso corpus, ma non con un altro).

Conclusioni

- Un tagger che implementi un algoritmo di ML ha il vantaggio di essere portabile: per ogni tipo di corpus da testare, è possibile fare il training iniziale in modo rapido, dato che le regole necessarie verranno apprese automaticamente.
- La sua accuratezza è comparabile a un tagger di tipo stocastico, il quale ha bisogno di migliaia di regole prima di estrarre informazioni contestuali: l'informazione infatti consiste in una lista di trigrammi che indicano per tutti i tag "tag_a, tag_b" la probabilità che tag_c *sia il successivo tag.*
- Un tagger come il Brill invece ha bisogno di circa 80 regole contestuali per annotare un testo correttamente. Le quali, inoltre, sono espresse in modo molto più comprensibile rispetto ad una lista di regole statistiche.

Webliografia

- **A Simple Rule-Based Part of Speech Tagger (Eric Brill)**
  - http://www.aclweb.org/anthology-new/A/A92/A92-1021.pdf
- **Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging (Eric Brill)**
  - http://acl.ldc.upenn.edu/J/J95/J95-4004.pdf
- **A Machine Learning Approach to POS Tagging (**Lluìs Màrquez, Lluìs Padró, Horacio Rodriguez)
  - http://www.springerlink.com/content/p8w6123l75766757/fulltext.pdf
- **Combining decision trees and transformation-based learning to correct transferred linguistic representations (**Simon Corston-Oliver and Michael Gamon**)**
  - http://www.mt-archive.info/MTS-2003-Corston.pdf
- **Acquiring disambiguation rules from text (Donald Hindle)**
  - http://www.aclweb.org/anthology-new/P/P89/P89-1015.pdf
- **Part-of-Speech Tagging (e lemmatizzazione**) (Baroni)
  - http://sslmit.unibo.it/~baroni/compling04f/pos.pdf
- **Lezioni num. 12 e 13 del Prof. Alessandro Lenci**