Come funziona il Part-of-Speech Tagging

Marco Baroni

14 marzo 2005

1 Introduzione

- Abbiamo usato taggers "pre-addestrati" che, dato un corpus in input, producono output taggato.
- In questo handout, studiamo come e perché il Part-of-Speech Tagging funziona, prendendo come esempio il Markov model tagging, che è tra i metodi più classici, più diffusi e con performance migliore.

2 POS tagging e apprendimento supervisionato

- Invece di formulare esplicitamente regole per tagging:
  - annota a mano un campione di dati (training corpus);
  - lascia che programma estragga generalizzazioni da dati annotati;
  - usa programma "addestrato" su dati annotati per annotare nuovi dati;
- In linea di massima, annotare a mano è un task più facile che formulare a mano le generalizzazioni appropriate.
- Ovviamente, taggers funzioneranno meglio se usati per taggare corpora simili a training corpus (e.g., un tagger allenato su dati presi dalla Repubblica funzionerà meglio se usato per taggare il Corriere della Sera che se usato per taggare la Divina Commedia).

2.1 Training corpus vs. test corpus

- Tagger non va valutato sugli stessi dati che sono stati usati per addestramento!
- Conta la capacità di *generalizzare* di tagger, dunque esso va valutato su dati che non ha visto in fase di addestramento.
- Potenziali pericoli di valutazione su training corpus:

1. Sopravvalutazione di performance (e.g., tagger riconosce una parola rara nel training corpus perché l'ha vista durante il training, ma non riconoscerebbe una parola altrettanto rara in un altro corpus).
2. Si favoriscono taggers che tendono all'"overtraining", cioè a favorire soluzioni *ad hoc* che funzionano molto bene nel training corpus ma non è detto si possano generalizzare (e.g., se in training corpus la parola *torta* è sempre seguita dalla parola *Sacher*, un tagger che predice che la parola *torta* debba sempre essere seguita da nome proprio funzionerà molto bene in training corpus, ma probabilmente non in altri corpora.)

- Dunque, di solito si mette da parte una certa percentuale dei dati annotati a mano, che non vengono usati per addestrare il tagger ma soltanto per valutarne la performance (testing corpus).

3 Markov Model Tagging

- Tutti i taggers basati sul paradigma del supervised learning estraggono statistiche sull'occorrenza di certi patterns nel training corpus, e poi usano tali statistiche per taggare nuovi dati.
- Tuttavia, i taggers differiscono, anche notevolmente, nelle statistiche che raccolgono e come, e in come applicano tali statistiche a nuovi dati.
- Markov model tagging è uno dei modelli più intuitivi, molto diffuso e con ottima performance.
- Il primo (?) Markov model tagger: Church 1988.
- Alcune implementazioni recenti: TnT (Brants 2000), ACOPOST tt (Schroeder 2002), FreeLing (Carreras, Chao, Padró & Padró 2004).
- Interpretazione di problema di tagging in termini di teoria della probabilità.
- Data una sequenza di parole $w_1, ..., w_n$, vogliamo trovare la sequenza di tags $t_1, ..., t_n$ che ha la probabilità più alta.

$$t_1, ..., t_n = \underset{t_1,...,t_n}{\text{argmax}} P(t_1, ..., t_n | w_1, ..., w_n) \tag{1}$$

- Adesso, passo per passo, scomporremo questa equazione in una serie di termini che sappiamo come calcolare.
- Usando la legge di Bayes, possiamo riscrivere l'equazione (1) come:

$$t_1, ..., t_n = \underset{t_1,...,t_n}{\text{argmax}} \frac{P(w_1, ..., w_n | t_1, ..., t_n) P(t_1, ..., t_n)}{P(w_1, ..., w_n)} \tag{2}$$

- Notate ora che il denominatore di (2) sarà costante per tutte le combinazioni di tags (poiché la sequenza di parole non varia), e dunque non avrà alcun effetto sul calcolo di quale sequenza di tags è più probabile.
- Possiamo dunque semplificare (2) togliendo il denominatore:

$$t_1, ..., t_n = \underset{t_1,...,t_n}{\text{argmax}} P(w_1, ..., w_n | t_1, ..., t_n) P(t_1, ..., t_n) \tag{3}$$

- Due assunzioni che ci semplificano la vita:
  1. La probabilità di una parola dipende solo dalla propria tag, non dalle tags delle altre parole nella frase:

$$P(w_1, ..., w_n | t_1, ..., t_n) \approx P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)$$

  2. *Assunzione markoviana* (del primo ordine):

$$P(t_1, ..., t_n) \approx P(t_1|t_0)P(t_2|t_1)...P(t_n|t_{n-1})$$

- Date queste assunzioni, (3) diventa:

$$t_1, ..., t_n = \underset{t_1,...,t_n}{\text{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}) \tag{4}$$

- Il secondo termine di (4) è dato dal prodotto delle probabilità, per ciascuna tag t_i, di incontrare la tag t_i se la tag immediatamente precedente è t_{i-1}.
- Il primo termine in (4) è dato dal prodotto delle probabilità, per ciascuna coppia parola/tag w_i/t_i, di incontrare la parola w_i se sappiamo che la tag corrispondente è t_i.
- In parole, il secondo termine risponde alla domanda: quanto è plausibile la tag t_i se ho appena visto la tag t_{i-1}? (E.g., quanto è plausibile che la tag della parola che sto analizzando sia VER:pper se la parola che precede ha la tag AUX:fin?)
- Il primo termine risponde alla domanda: quanto è plausibile che io veda la parola w_i se so che tale parola ha la tag t_i? (E.g., quanto è plausibile che la parola che sto analizzando sia *fatto* se so che è un participio passato verbale?)
- In fase di tagging, un'applicazione ingenua di (4) a una frase richiederebbe il calcolo della formula per tutte le possibili combinazioni di tags.
- Fortunatamente, esistono algoritmi molto efficienti per trovare rapidamente la sequenza di tags che produce il valore più alto della formula in questione.

3.1 Training

- Avendo ridotto la formulazione probabilistica astratta del problema del tagging presentata in (1) a (4), ci troviamo con una serie di fattori facili da stimare sulla base di un training corpus annotato.
- Le probabilità dei fattori del primo termine di (4) possono essere stimate così:

$$\hat{P}(w_i|t_i) = \frac{fq(t_i, w_i)}{fq(t_i)}$$

- Le probabilità dei fattori del secondo termine possono essere stimate così:

$$\hat{P}(t_i|t_{i-1}) = \frac{fq(t_{i-1}, t_i)}{fq(t_{i-1})}$$

3.2 Esempio

- Una frase altamente ambigua:

```
l'ho fatto
```

- Usando l'equazione (4), le formule nella sezione 3.1 per la stima delle probabilità e i "training data" dal corpus la Repubblica riportati nella Tabella 1, calcolare quale sequenza di tags risulta più probabile tra:

PRO:pers	AUX:fin	VER:pper
ART	AUX:fin	VER:pper

- (Potete provare anche altre combinazioni, se siete in vena.)
- Da notare:
  - Nel calcolo di $P(t_i|t_{i-1})$ per l'elemento in inizio frase, considerate SENT come t_{i-1}.
  - Se vogliamo trovare la sequenza più probabile tra le due indicate sopra, dobbiamo proprio calcolare tutte le 6 probabilità richieste da (4)? Quali sono le probabilità che vanno davvero calcolate?
  - Quale fattore rende la prima sequenza più probabile della seconda?

3.3 Due problemi per il tagging

- Due problemi per i Markov model taggers, che però si pongono in una maniera o nell'altra con tutti i taggers:
  - Che probabilità assegniamo alle sequenze di tags che non capitano mai in training corpus?
  - Che tag assegniamo alle parole che non capitano mai in training corpus?

elemento/i	frequenza
SENT PRO:pers	335867
PRO:pers AUX:fin	1049068
AUX:fin VER:pper	5744740
SENT ART	2827575
ART AUX:fin	127551
PRO:pers VER:fin	2651538
VER:fin VER:pper	228886
AUX:fin NOM	66493
ART l'	3292774
PRO:pers l'	441315
VER:fin ho	35973
AUX:fin ho	169960
NOM fatto	120654
VER:pper fatto	201535
SENT	13893432
PRO:pers	4882771
AUX:fin	7406176
VER:pper	11640259
ART	30315898
VER:fin	17470049
NOM	67441308

Tabella 1: Frequenze estratte dal corpus la Repubblica (con aggiustamenti).

3.3.1 Sequenze di tags sconosciute

- Se tagset è esteso e/o training corpus piccolo e/o usiamo modello markoviano di ordine maggiore (e.g., trigrammi invece di bigrammi), molte sequenze di tags saranno estremamente rare o inesistenti nel training corpus: problema per stima di probabilità.
- Spesso stime più robuste ottenute con *interpolazione lineare*:

$$P(t_i|t_{i-2}, t_{i-1}) \approx \lambda_1 \hat{P}(t_i|t_{i-2}, t_{i-1}) + \lambda_2 \hat{P}(t_i|t_{i-1}) + \lambda_3 \hat{P}(t_i)$$

- Per es.:

$$P(\text{VER}|\text{PRO}, \text{AUX}) \approx$$
$$\lambda_1 \hat{P}(\text{VER}|\text{PRO}, \text{AUX}) + \lambda_2 \hat{P}(\text{VER}|\text{AUX}) + \lambda_3 \hat{P}(\text{VER})$$

3.3.2 Parole sconosciute

- Stima di $P(w_i|t_i)$ richiede computo di frequenza in training corpus di parole specifiche.
- Nessun training corpus, per quanto esteso, contiene tutte le parole possibili in una lingua.
- Problemi con: termini tecnici, forme flesse, prestiti, neologismi, nomi propri, marche...

- Serve metodo per stimare $P(w_i|t_i)$ di parole non in training corpus.
- Il metodo di maggior successo si basa su *analisi dei suffissi*.
- $P(t|astonizzerebbero)$ stimato usando combinazione di $\hat{P}(t|...zzerebbero)$, $\hat{P}(t|...zerebbero)$, ... , $\hat{P}(t|...ebbero)$, ... , $\hat{P}(t|...ro)$, $\hat{P}(t|...o)$.
- $P(astonizzerebbero|t)$ derivabile da $P(t|astonizzerebbero)$ via legge di Bayes.
- Più in generale: per lingua con morfologia ricca come italiano, ha senso integrare tagger con ampio lessico morfologico e analisi morfologica automatica.

4 Altri tipi di taggers

- Transformation-based learning (Brill 1995): comincia con tagging "naive" di input (e.g., assegnando a ciascuna parola la sua tag più frequente) e poi corregge tagging naive con regole di correzione estratte dal training corpus.
- Per esempio, transformation-based learning tagger addestrato su corpus la Repubblica ha estratto regole quali: *se hai taggato una parola come verbo ma quella che segue è anche un verbo, ri-tagga la parola come ausiliare.*
- Decision trees (Schmid 1994): costruzione automatica di "test" per decidere tag più probabile di parola w in contesto X.
- Per esempio:
  1. Parola immediatamente a sinistra è aggettivo? Sì: vai a 2.; No: vai a 3.
  2. Parola immediatamente a destra è verbo? Sì: fermati e assegna tag Nome; No: vai a...
  3. ...
- Maximum entropy (Ratnaparkhi 1996): Un modello che può tenere conto e integrare in maniera ottimale un gran numero di indizi potenzialmente rilevanti, sia simili tra di loro che di diversa natura.

5 Performance

- State of the art: accuratezza attorno al 97.5% per l'inglese, tipicamente un po' più bassa per altre lingue (per l'italiano, noi abbiamo raggiunto 95.5% sul corpus la Repubblica).
- Accuratezza del 97.5% vuol dire che tagger assegna stessa tag assegnata da esperto a 97.5% di parole in test set.

- Ma... "there are lies, damned lies, and statistics"(Abney):
  - Percentuale di *frasi* che contengono errori molto più alta!
  - Tagger naive in inglese ha già accuratezza attorno al 90% (tagger naive: assegna sempre a una parola la tag che ha più di frequente in training corpus; e.g., tratta sempre *l'* come articolo).
- Le combinazioni di taggers funzionano meglio dei taggers singoli.
- Tra i taggers singoli, quelli con performance più alta tendono ad essere il Markov model tagger (a patto che abbia una buona componente per la gestione delle sequenze non incontrate nel training corpus e delle parole sconosciute) e il maximum entropy tagger.