Part-of-Speech Tagging (e lemmatizzazione)

Marco Baroni

3 dicembre 2004

1 Introduzione

- Lo scopo: associare ciascun token in un corpus ad una parte del discorso.

```
Il/ART cane/N abbaia/V ./PUNT

Il ART:m:s
cane N:m:s
abbaia V:ind:pr:3:s
. PUNT:sent
```

- Perché?
- Sorprendentemente utile:
  - Ricerche linguistiche
  - Estrazione di termini complessi, collocazioni, ecc.
  - Named entity recognition
  - Modelli predittivi per correttori ortografici, speech recognition e simili
  - Punto di partenza per quasi tutte le applicazioni linguistiche più complesse: parsing, information extraction, traduzione automatica, ecc.
- Sorprendentemente difficile, anche in lingua con morfologia ricca come italiano.

```
people like you have good taste
una vecchia porta la sbarra
perdono
sempre secondo fonti dei servizi
della criminalita'
gli devo molto
ha portato il suo
di cui hai dato prova nel proseguire
```

- In quasi tutti i casi, disambiguazione è resa possibile da contesto e/o indizi morfologici (maggior enfasi su contesto, probabilmente per una questione di anglocentrismo).
- Per corpora di dimensioni non risibili, non è realistico condurre annotazione manuale.
- Annotazione automatica, di solito con metodi che rientrano in paradigma di *apprendimento supervisionato.*

2 POS tagging e apprendimento supervisionato

- Supervised learning: approccio molto comune in machine learning e linguistica computazionale, non limitato a tagging, anche se tagging ne è esempio classico.
- Invece di formulare esplicitamente regole per tagging:
  - annota a mano un po' di dati;
  - 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.

2.1 Training, testing e tagging

- La situazione: abbiamo un tagger (il programma da addestrare) e un corpus che vogliamo "taggare" (etichettare??)
- La procedura tipica:
  - Decidi *tagset* (lista di parti del discorso da usare).
  - Scegli subset di corpus da annotare a mano (meglio se random subset), e annotalo.[1]
  - Dividi dati in subset in due parti: la maggioranza dei dati va in *training set*, ma lascia un po' di dati per *test set* (e.g., usa 90% per training e 10% per testing).
  - Addestra tagger su training set.
  - Valuta performance del tagger su test set.
  - Se performance è soddisfacente, usa tagger addestrato per taggare il resto del corpus.

[1] Dimensioni tipiche della parte annotata a mano: per l'inglese, ∼1M di tokens; per altre lingue, qualche centinaia di migliaia di tokens (per il corpus la Repubblica/SSLMIT, training su più o meno 110K tokens).

- NB: per molti scopi, se tagger pre-addestrato è disponibile, annotazione manuale, training e testing non sono necessari.

2.2 Training set vs. test set

- 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 dati già visti:
  1. Sopravvalutazione di performance (e.g., un tagger magari riconosce tutti i nomi propri in 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 (e.g., se in training set 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 training set, ma probabilmente non con un altro corpus).

3 Tipi di taggers

- 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.
- Qui, faccio una brevissima rassegna di alcuni modelli, concentrandomi su due schemi (Markov model e transformation-based) che sono un po' agli estremi opposti della tipologia di taggers.

3.1 Baseline tagger

- Assegna a ciascuna parola la tag a cui è associata più di frequente in training corpus.
- Usato principalmente come termine di paragone per modelli più sofisticati.
- In inglese, ha accuratezza attorno al 90%.

3.2 Markov model taggers

- Il modello più semplice ed intuitivo, molto diffuso.
- Il primo (?) Markov model tagger: Church 1988.
- Alcune implementazioni recenti: TnT (Brants 2000), ACOPOST tt (Schroeder 2002), FreeLing (Carreras, Chao, Padró & Padró 2004).

3.2.1 Training

- In trigram Markov model taggers (il tipo più comune) probabilità di tag data da:

$$P(t_n) = P(w_n|t_n)P(t_n|t_{n-2}, t_{n-1})$$

- Esempio:

```
Il/ART cane/N abbaia
```

- Probabilità che ultima tag sia V data da:

$$P(t_3 = \text{V}) = P(\text{abbaia}|\text{V})P(\text{V}|\text{ART,N})$$

- Training corpus usato per stimare le probabilità. Per esempio:

$$\hat{P}(\text{abbaia}|\text{V}) = \frac{fq(\text{abbaia, V})}{fq(\text{V})}$$

$$\hat{P}(\text{V}|\text{ART,N}) = \frac{fq(\text{ART N V})}{fq(\text{ART N})}$$

- 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?

3.2.2 Sequenze di tags sconosciute

- Specialmente se tagset è esteso (e/o training corpus piccolo), molti trigrammi saranno estremamente rari o inesistenti nel training corpus: problema per stima di probabilità.
- Spesso (per es., in TnT) stime più robuste ottenute con *interpolazione lineare*:

$$P(t_n|t_{n-2}, t_{n-1}) = \lambda_1 \hat{P}(t_n|t_{n-2}, t_{n-1}) + \lambda_2 \hat{P}(t_n|t_{n-1}) + \lambda_3 \hat{P}(t_n)$$

- Per es.:

$$P(\mathrm{V}|\mathrm{ART}, \mathrm{N}) = \lambda_1 \hat{P}(\mathrm{V}|\mathrm{ART}, \mathrm{N}) + \lambda_2 \hat{P}(\mathrm{V}|\mathrm{N}) + \lambda_3 \hat{P}(\mathrm{V})$$

3.2.3 Parole sconosciute

- Stima di $P(w|t)$ 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: forme flesse, prestiti, neologismi, nomi propri, marche...
- Serve metodo per stimare $P(w|t)$ di parole non in training corpus.
- Il metodo di maggior successo, adottato anche da TnT, 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 formula di Bayes.
- Più in generale: per lingua con morfologia ricca come italiano, ha senso integrare tagger con ampio lessico morfologico.

3.2.4 Tagging

- Semplicemente:

$$t_1...t_n = \underset{t_1...t_n}{argmax} \prod_{i=1}^{n} P(t_i)$$

- Ovvero: per ogni frase scegli la sequenza di tags che assegna alla frase la probabilità più alta, dove la probabilità della frase è data dal prodotto delle probabilità delle tags.
- Esistono algoritmi efficienti per trovare la sequenza $t_1...t_n$ che massimizza $\prod_{i=1}^{n} P(t_i)$

3.3 Transformation-based learning

- Brill 1995.
- In poche parole:
  - Training corpus annotato usando algoritmo "naive" (per es., baseline method)
  - Regole di correzione estratte automaticamente paragonando corpus annotato da algoritmo naive a training corpus annotato a mano.
  - Algoritmo naive e regole di correzione usate per tagging.

3.3.1 Training

1. Crea una versione del training corpus senza tags (corpus da annotare) e un lessico che contenga per ciascuna parola la tag che ha più di frequente nel training corpus.
2. Annota ciascuna parola nel corpus da annotare con tag che ha nel lessico, o con N se non in lessico.
3. Prova ad applicare, una alla volta, una serie di regole di correzione al corpus così annotato, e scegli la regola che corregge il maggior numero di errori nel corpus da annotare, paragonato al training corpus originario.
4. Applica la regola scelta al punto precedente al corpus da annotare.
5. Ripeti 3. e 4. fino a quando il numero di errori corretti dalla miglior regola diventa inferiore ad una soglia ϵ.

3.3.2 Regole di correzione

- Automaticamente costruite dall'algoritmo sulla base di ristretto numero di templates che specificano set di contesti/caratteristiche potenzialmente rilevanti.
- Le regole più comuni estratte nel training su dati la Repubblica/SSLMIT (nel formato del transformation-based tagger della suite ACOPOST):

```
NOM rare[0]
NPR rare[0] tag[0]=NOM cap[0]=some
ADJ tag[-1]=NOM tag[0]=NOM
AUX:fin tag[0]=VER:fin tag[1]=VER:pper
VER:infi rare[0] tag[0]=NOM suffix[0]=re
VER:fin rare[0] suffix[0]=no
VER:pper tag[-1]=AUX:fin tag[0]=NOM
ADJ tag[0]=NOM tag[1]=NOM
AUX:fin tag[0]=VER:fin tag[1]=AUX:pper
VER:geru rare[0] suffix[0]=ndo
```

- Per esempio, la terza regola si legge così: se la tag che precede è NOM e la tag che stiamo considerando è NOM, allora cambia quest'ultima in ADJ.

3.3.3 Tagging

- Assegna a ciascuna parola in corpus tag che ha in lessico.
- Applica regole estratte durante il training nell'ordine in cui sono state estratte (dalla più generale alla più specifica).

3.4 Altri modelli

- Decision trees (TreeTagger, 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 che di diversa natura.
- Memory-based (Daelemans et al. 1996): Mantiene un vasto database degli esempi incontrati nel training corpus, e assegna tags sulla base di esempio in database più "simile" a contesto corrente.

3.5 Combinazione di taggers

- Spesso la performance migliore si ottiene combinando l'output di vari taggers.
- Il tipo di combinazione più semplice: per ciascuna parola, scegli la tag che le è stata assegnata dal numero maggiore di taggers (majority voting).

4 Performance

- State of the art: accuratezza attorno al 97.5% (per l'inglese).
- Cioè: 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!
  - Baseline in inglese ha accuratezza attorno al 90%.

- Performance di taggers singoli in inglese (LOB, WSJ) e olandese (Wotan, WotanLite), da van Halteren et al. 2001:

	LOB	WSJ	Wotan	WotanLite
baseline	93.22	94.57	89.48	93.40
markov model	97.55	96.63	92.06	95.26
transformation-based	96.37	96.28	-	94.63
maximum entropy	97.52	96.88	91.72	95.56
memory-based	97.06	96.41	89.78	94.92

- Combinazioni:

	LOB	WSJ	Wotan	WotanLite
voting	97.76	96.98	92.51	96.01
stacking	98.14	97.23	93.03	96.42

- Questi dati rispecchiano tendenze generali, almeno per l'inglese, ossia:
  - 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.

- Tuttavia, altri taggers hanno altri vantaggi: per esempio, le regole generate dal transformation-based tagger sono facili da capire e magari manipolare a mano.

- I nostri risultati con il corpus la Repubblica/SSLMIT (10-fold cross validation):

markov model	95.04
transformation-based	95.20
memory-based	93.95
voting	95.62
stacked voting	95.61

5 Tagsets

- `Il ART cane N abbaia V . PUNT`

- `Il ART:m:s cane N:m:s abbaia V:ind:pr:3:s . PUNT:sent`

- Maggiore granularità aiuta se nuove tags sono sensate dal punto di vista distribuzionale.

- `ha V cantato V` vs. `ha AUX cantato V`
- Ma più tags significa frequenze minori per ciascuna tag, e maggiori problemi di data sparseness (e.g., per avere statistiche significative per una tag quale `V:cong:impf:3:pl` occorre probabilmente avere un training corpus enorme).
- Tagset del corpus la Repubblica/SSLMIT:

tag	*description*	*tag*	*description*
ADJ	adjective	ADJ:abr	adjectival abbreviation
ADV	adverb	ADV:abr	adverbial abbreviation
ART	article	ASP:fin	aspect. verb fin. form
AUX:fin	aux. verb fin. form	AUX:geru	aux. verb gerundive
AUX:infi	aux. verb infinitive	AUX:pper	aux. verb past part.
CAU:fin	caus. verb fin. form	CAU:geru	caus. verb gerund.
CAU:infi	caus. verb infinitive	CAU:pper	caus. verb past part.
CLI:ne	ne clitic	CLI:si	si clitic
CON:coo	coordinating conj.	CON:sub	subordinating conj.
DET:demo	demonstrative det.	DET:indef	indefinite determiner
DET:num	numeral determiner	DET:poss	possessive determiner
DET:wh	wh determiner	INT	interjection
LOA	loan word	MOD:fin	modal verb fin. form
MOD:infi	mod. verb infinitive	MOD:pper	mod. verb past part.
NOM	noun	NOM:abr	nominal abbreviation
NPR	proper noun	NPR:abr	proper noun abbrev.
NUM	number	PON	punctuation mark
PRE	preposition	PRE:art	prep. with article
PRO:demo	demonstrative pron.	PRO:indef	indefinite pronoun
PRO:num	numeral pronoun	PRO:pers	personal pronoun
PRO:poss	possessive pronoun	PRO:wh	wh pronoun
SENT	sentence marker	UNK	unknown
VER:fin	verb finite form	VER:geru	verb gerundive
VER:infi	verb infinitival	VER:pper	past participle
VER:ppre	present participle	WH	wh element
WH:che	che		

6 La lemmatizzazione

- La lemmatizzazione consiste nell'associare ad ogni parola un lemma:

```
i         il
cani      cane
abbaiano  abbaiare
```

- Spesso gli stessi programmi usati per il tagging compiono anche la lemmatizzazione di un corpus.
- La lemmatizzazione è quasi sempre basata su un dizionario che associa un lemma ad ogni coppia parola/tag (dunque, lemmatizzazione ha luogo dopo tagging[2]):

[2] Anche se a sua volta tagging sarà basato su qualche forma di dizionario.

```
porta  NOM     porta
porta  VER:fin portare
```

- Lemmatizzatore può anche avere componente che cerca di indovinare lemma di parole non in dizionario, per es. attraverso regole di (de)suffissazione (e.g., `izzerebbero -> izzare`).

7 Usare un tagger pre-addestrato

- Se esiste un tagger pre-addestrato per la lingua su cui stiamo lavorando, spesso fare il tagging di un corpus è solo questione di un singolo, semplice comando (e.g., `tree-tagger-german < corpus > corpus.tgd`).
- Alcuni punti da considerare:
  - Tagset è adeguato ai nostri scopi? (e.g., se vogliamo studiare la differenza tra nomi singolari e plurali, sarebbe meglio usare un tagger il cui tagset distingue tra queste categorie)
  - Performance è adeguata ai nostri scopi? (e.g., performance richiesta per estrazione terminologica è probabilmente inferiore a performance necessaria per sistema di traduzione automatica)
  - La natura dei dati nel nostro corpus è ragionevolmente simile alla natura dei dati usati per allenare il tagger? (e.g., un tagger allenato su la Repubblica funzionerà meglio se usato per taggare il Corriere della Sera che se usato per taggare la Divina Commedia...)
  - Che tipo di input si aspetta il tagger? (e.g., il TreeTagger per l'italiano vuole *perchè* al posto di *perché*, e in generale non ama le tabs...)

8 I taggers che useremo per i nostri corpora

8.1 Inglese e tedesco

- TreeTagger (decision tree tagger):

```
$ tree-tagger-english < corpus > corpus.tgd
$ ... | tree-tagger-english > corpus.tgd
$ tree-tagger-german < corpus > corpus.tgd
$ ... | tree-tagger-german > corpus.tgd
```

- Output in formato: PAROLA TAG LEMMA

8.2 Italiano

- TreeTagger (decision tree tagger), ma usando un wrapper che corregge alcuni problemi di tagging e lemmatizzazione.

```
$ ita_tree_tagger_wrapper.pl corpus > corpus.tgd
$  ... | ita_tree_tagger_wrapper.pl - > corpus.tgd
```

- Output in formato: PAROLA TAG LEMMA

8.3 Spagnolo

- FreeLing analyzer (Markov model tagger: Carreras, Chao, Padró & Padró 2004).
- Effettua anche il riconoscimento di nomi propri, quantità, date ecc., a meno di non bloccare tali funzionalità usando le opzioni mostrate nel secondo esempio.

```
$ analyzer -f /usr/local/share/FreeLing/config/es.cfg --outf tagged \
< corpus > corpus.tgd
$ ... | analyzer -f /usr/local/share/FreeLing/config/es.cfg --outf \
tagged > corpus.tgd
$ analyzer -f /usr/local/share/FreeLing/config/es.cfg --noloc --nonumb \
--nodate --noquant --noner --nonec --outf tagged < corpus > corpus.tgd
```

- Output in formato: PAROLA LEMMA TAG

8.4 Alcune cose a cui stare attenti

- L'input è in un formato che il tagger gestisce correttamente? (Per esempio, il tagger italiano riconosce correttamente le parole con l'apostrofo?)
- Ci sono caratteri speciali che fanno impazzire il tagger? (Per esempio, la tab?)
- Tutti problemi che si risolvono facilmente con un po' di `sed`...
- Nell'output, ogni riga ha tre e solo tre campi? Siccome l'analisi dell'output si baserà probabilmente sull'assunzione che sia così, è meglio verificare che non ci siano righe con più o meno campi (`$ gawk 'NF != 3' corpus.tgd | more`) e, dopo aver ispezionato tali righe, eliminarle (`$ gawk 'NF == 3' corpus.tgd > cleaned.corpus.tgd`), o, se piu' appropriato, ripulire l'input e ritaggare.[3]

[3] I TreeTagger ignorano le stringhe di forma <.*> che non contengono spazi, ossia le stampano così come sono nell'output, senza lemma e senza tag. Questa è dunque una tipica fonte di righe con meno o più di tre campi.

9 Per saperne di più...

- La mia pagina dei links ha collegamenti alle pagine del TreeTagger, di FreeLing e di altri part-of-speech taggers.
- Tutte le introduzioni alla linguistica computazionale dedicano ampio spazio al part-of-speech tagging, per es.:
  - Charniak 1993
  - Manning e Schütze 1999
  - Jurafsky e Martin 2000
- Articoli
  - A carattere introduttivo: Abney 1996
  - Su taggers specifici: Brants 2000, Brill 1995, Ratnaparkhi 1996, Schmid 1994
  - Tagger comparison & combination: van Halteren, Zavrel e Daelemans 2001
  - The cutting edge: Cucerzan e Yarowsky 2002
  - E, naturalmente, molti altri ancora!

Bibliografia minima

S. Abney. 1996. Part-of-speech tagging and partial parsing. In Church, Young e Bloothooft (a cura di). *Corpus-based methods in language and speech.* Dordrecht: Kluwer.

M. Baroni, S. Bernardini, F. Comastri, L. Piccioni, A. Volpi, G. Aston e M. Mazzoleni. 2004. Introducing the La Repubblica corpus: A large, annotated, TEI(XML)-compliant corpus of newspaper Italian. *Proceedings of LREC* 2004.

T. Brants. 2000. TnT: A statistical part-of-speech tagger. *ANLP* 2000.

E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* 21: 543-565

E. Brill e J. Wu. 1998. Classifier combination for improved lexical disambiguation. *COLING-ACL* 1998: 191-195

X. Carreras, I. Chao, L. Padró e M. Padró. 2004. FreeLing: An open-source suite of language analyzers. *Proceedings of LREC* 2004.

E. Charniak. 1993. *Statistical language learning.* Cambridge: MIT Press.

K. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. *ANLP* 1988: 136-143

M. Civit, M. Martí e L. Padró. 2003. Using hybrid probabilistic-linguistic knowledge to improve pos-tagging performance. *Corpus Linguistics* 2003: 810-817.

S. Cucerzan e D. Yarowsky. 2000. Language independent minimally supervised induction of lexical probabilities. *ACL* 2000: 270-277

S. Cucerzan e D. Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. *CoNLL* 2002.

R. Garside, G. Leech e A. McEnery (a cura di). 1997. *Corpus annotation*. London: Longman.

B. Greene e G. Rubin. 1971. *Automatic grammatical tagging of English*. Brown University technical report.

J. Hajič. 2000. Morphological tagging: data vs. dictionaries. *ANLP-NAACL* 2000: 94-101

D. Jurafsky e J. Martin. 2000. *Speech and language processing*. Upper Saddle River: Prentice-Hall.

C. Manning e H. Schütze. 1999. *Foundations of statistical natural language processing*. Cambridge: MIT Press.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. *EMNLP* 1996.

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. *International Conference on New Methods in Language Processing*.

H. Schmid. 1995. Improvements in part-of-speech tagging with an application to German. *EACL-SIGDAT* 1995.

I. Schröder. 2002. A case study in part-of-speech tagging using the ICOPOST toolkit. *Computer Science Memo* 314/02, Department of Computer Science, University of Hamburg.

F. Tamburini. 2000. Annotazione grammaticale e lemmatizzazione di corpora in italiano. In Rossini Favretti (a cura di). *Linguistica e informatica*. Roma: Bulzoni: 57-73

H. van Halteren, J. Zavrel e W. Daelemans. 2001. Improving accuracy in word class tagging through combination of machine learning systems. *Computational Linguistics* 27: 199-229