Relazione TLN

Carlo Alberto Barbano
Mat. 811588

1 - Traduttore diretto EN-IT

1.1 - TreeBank

Per la creazione dei modelli è stato utilizzato il treebank Universal Dependency nella versione inglese. Per la gestione del dataset è stata creata la classe TreeBank (treebank.py) che si occupa di parsificare i treebank e di offrire un'interfaccia agevole per il loro utilizzo.

La classe TreeBank precalcola inoltre le probabilità di emissione delle parole per ogni tag e il numero totale di occorrenze dei tag, memorizzando le informazioni in un dizionario (tabella di hash) in modo da ottenerle in un tempo O(1).

Es.

```
#probabilità di emissione della parola "make" per il tag VERB
treebank.tags['VERB']['emission']['make']
#numero di occorrenze del tag VERB nel treebank
treebank.tags['VERB']['count']
```

Per accedere ai dati del treebank, la classe TreeBank può essere utilizzata come un iteratore:

```
for sentence, tags in treebank:
    # sentence is a list of tokens
    # tags are the pos tag for each token in the sentence
    …
```

1.1 - PoS Tagging

Per la prima parte del progetto sono stati sviluppati due modelli di PoS Tagger: un modello di baseline (BaselinePoSTagger) e un modello HMM (MarkovPoSTagger).

Per addestrare un modello, è sufficiente effettuare una chiamata al metodo *`train`*, passando come argomento un treebank. La procedura di training si occupa di calcolare le probabilità necessarie per la fase di decoding.

Baseline Tagger

Il modello baseline assegna il tag più frequente per una certa parola all'interno del treebank, e il tag NOUN nel caso di parole sconosciute. L'accuracy è di 89% sul test set e 94% sul training set.

Durante il training, il modello baseline calcola la frequenza con cui ogni tag è associato a una certa parola, e in fase di decoding selezionerà per ogni parola il tag con frequenza maggiore.

HMM Tagger

Il secondo modello individua la sequenza di tag più probabile per una certa sequenza di parole, usando l'algoritmo di Viterbi. L'accuracy ottenuta è di 91% sul test set e 96% sul training set.

Nel training, il modello HMM calcola le probabilità di transizione da un tag *t* a un tag *t+1*, e in fase di decoding usa questo risultato, insieme alla probabilità di emissione dei tag (calcolata dal treebank) per produrre la sequenza di tag usando l'algoritmo di Viterbi:

$$v_t(j) = max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

dove:

- $v_t(i)$ è il valore di viterbi calcolato al passo *t* per lo stato *i*
- a_{ij} è la probabilità di transizione dallo stato *i* allo stato *j*
- $b_j(o_t)$ è la probabilità di emissione per il token o_t nello stato b_j

Nel caso in cui una transizione da uno stato *i* a uno stato *j* non fosse stata osservata nel treebank di training, viene effettuata un'ipotesi di smoothing iniziale

$$\alpha = P(t_j|t_i) = 1 / \#tags$$

Per quanto riguarda la probabilità di emissione $b_j(o_t)$, nel caso in cui la parola o_t fosse sconosciuta vengono effettuate le seguenti ipotesi iniziali di smoothing:

$$P(o_t|b_j) = \begin{cases} \beta = 1. & b_j = PNOUN \ and \ isCapitalized(o_t) \\ \gamma = 1./\#tags & otherwise \end{cases}$$

Per l'ottimizzazione degli iperparametri $\alpha\ \beta\ \gamma$ viene effettuata una grid search in un range $[1.0;\ 1e-10]$ usando il development set, con una chiamata al metodo `MarkovPoSTagger.tune_hyperparameters(dev_set)`. I valori ottimali degli iperparametri trovati in questo modo sono:

$$\alpha = 0.01;\ \beta = 1e-6;\ \gamma = 1e-7;$$

1.2 - Traduttore

La fase di traduzione è implementata dalla classe Translator. L'algoritmo di traduzione è diviso in tre fasi:

1. PoS Tagging
2. PoS reordering
3. Traduzione

PoS Tagging:

Per la prima fase viene utilizzato uno dei modelli descritti in precedenza: data una frase in input, ne viene prodotta una tokenizzazione usando la libreria nltk, usata come input per il PoS tagger.

Reordering:

Una volta prodotti i tag per la frase, vengono applicate le regole di reordering da inglese a italiano. Le regole sono descritte in un file apposito (*rules.txt*) e si presentano in una forma del tipo

```
<TAGS SEQUENCE (EN)> <ORDER (IT)>
```

Esempio:

```
AUX PART   1    0
#are not   not  are
```

La parte sinistra (testa) della regola contiene la sequenza di tags da matchare, la parte destra (corpo) contiene gli indici dei tag nella sequenza in cui devono essere riscritti (nell'esempio precedente *index(AUX)=0 index(PART)=1*).

Le regole inoltre possono essere specializzate su token desiderati, usando espressioni regolari, usando una forma del tipo TAG-{regex}

```
VERB       <- applica regola a qualsiasi tag VERB
VERB-.*ing <- applica regola a tag VERB il cui token termina con
"ing" (looking, going..)
```

La parte destra di ogni regola consente, oltre a specificare l'ordine di riscrittura dei tag, l'aggiunta, la rimozione e la sostituzione di token. Questo è utile nei casi di verbi fraseologici (es. *looking for*) o

Sostituzione:
Può essere ottenuta una sostituzione di un token, usando una forma del tipo
`index-{sostituzione}:`

Esempio:
```
PRON AUX-are VERB-.*ing          0 1-stai 2
```

Il verbo ausiliare "*are*" verrà sostituito con "*stai*" nei casi in cui è seguito da un verbo nella ing-form (e preceduto da un pronome come "*you*").

Rimozione:
E possibile rimuovere una coppia (TAG, token) con il carattere "~" usando una forma del tipo `index-~`.

Esempio:
```
VERB-looking ADP-for  0-cercando 1-~
```

Sostituzione+Rimozione: Il verbo fraseologico "looking for" verrà sostituito con "cercando", rimuovendo la coppia (ADP, for) dalla frase originaria.

Aggiunta:
Oltre a sostituire e rimuovere tag/token è possibile aggiungerne di nuovi. Questo serve nei casi in cui nella frase originale (inglese) sono stati omessi degli elementi (es. congiunzioni) che è necessario esplicitare nella traduzione italiana.
L'aggiunta di un token nella frase viene indicata con il carattere "+" (usato al posto dell'indice), dal token da aggiungere e dal PoS tag relativo.

Esempio:
```
DET NOUN PRON AUX VERB     0 1 +-che[SCONJ] 2 3 4
```

La frase
"The/DET droids/NOUN you/PRON are/AUX looking/VERB" diventa
"The/DET droids/NOUN **che/SCONJ** you/PRON are/AUX looking/VERB"

Inoltre è possibile effettuare un'applicazione parziale delle regole, per coprire più casi simili con una sola regola o evitare problemi dovuti da applicazioni ripetute di regole simili. Per indicare come facoltativa la parte finale di una regola è sufficiente prependere i tag (e i relativi indici) con il carattere punto ".".

Esempio:
```
PROPN PART-'s NOUN .CCONJ .NOUN 2 .3 .4 1 0
```

Il traduttore tenterà di applicare la regola estesa, rimuovendo la parte facoltativa in assenza di match.
Ad esempio la frase "*lowers Vader's mask and helmet onto [..]"* verrà riscritta come "*lowers mask and helmet 's Vader onto [..]"*, ma la stessa regola potrebbe essere applicata a una frase come "*lowers Vader's mask onto [..]"* producendo "*lowers mask 's Vader onto [..]*".

Traduzione

L’ultima fase consiste nella traduzione letterale delle parole. Per questo è stato prodotto un dizionario conentente un mapping 1-1 delle parole da inglese a italiano (*dict.txt*).

Per ottenere la traduzione di una frase, è sufficiente invocare il metodo `Translator.translate(sentence, tagger)` il cui valore di ritorno è costituito da una lista di tuple *(parola tradotta, pos tag)*.

2. Traduttore ITA-YODA