Relazione
Esercitazione 3

Hanks

Introduzione

La seguente esercitazione ha visto la definizione di un algoritmo che implementa la teoria di Patrick Hanks.

Per questa esercitazione è stato recuperato un corpus di 7.8 milioni di frasi (prese da *Wikipedia Sentences / Kaggle*) le quali contenevano il verbo transitivo scelto, ovvero **create.** In particolare, sono state estrapolate 1000 frasi dal corpus.

Successivamente, per ogni frase viene effettuato il parsing, e restituito il soggetto e complemento oggetto, relativi al verbo trattato.

Una volta trovati i possibili filler si è passati alla fase di disambiguazione degli stessi: per questo passaggio si è usato il metodo `lesk()` messo a disposizione dalla libreria *Python NLTK*.

Dopo aver effettuato la fase di disambiguazione si è passati alla fase di ricerca dei super-sensi. In questo caso si è fatto ricorso al metodo `lexname()` che restituisce il super-senso della parola a cui viene applicato. In questo modo sono state trovate le coppie di semantic types, successivamente fornite in output con le relative frequenze, rendendo facile la visualizzazione dei cluster

Struttura del codice ed implementazione

Per lo svolgimento di questa esercitazione il codice è diviso nei file

- **`main.py`**

  che si occupa di richiamare i metodi per l'esecuzione dell'algoritmo e di restituite i risultati attraverso un grafico.

- **`utils.py`**

  che implementa i seguenti metodi

- `parse_corpus()`
  metodo che si occupa di parsificare il coprus di frasi *wikisent2* restituendo una lista di 1000 frasi, scelte randomicamente, ma che rispettino i seguenti criteri:
  - la frase deve contenere il verbo d'interesse
  - il verbo in questione deve essere classificato come `VERB` (e non `ADJ`,`NOUN`, ecc..)
  - il verbo deve avere una dipendenza diretta al (deve essere legato) soggetto ed il complemento della frase

  Per eseguire questi controlli tale metodo si appoggia al altri due metodi `exist_subj_dobj(tokens)` e `verb_is_trans_verb (tokens,nsubj,dobj)`

- `exist_subj_dobj(tokens)`
  metodo per verificare se esiste un soggetto ed un complemento all'interno di una frase tokenizzata, passata in input

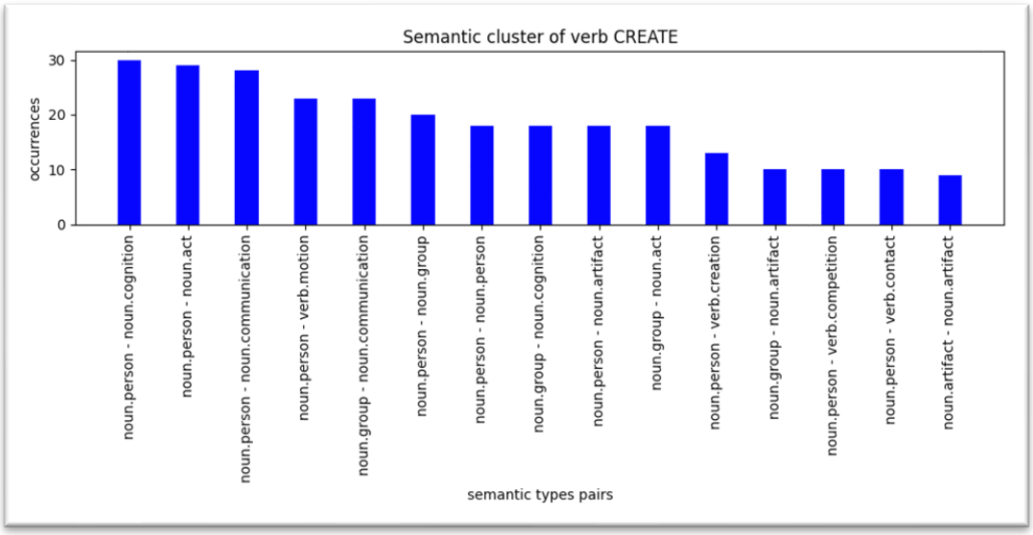
- `verb_is_trans_verb(tokens,nsubj,dobj)`
  metodo che prende in input la frase tokenizzata, un soggetto ed un complemento oggetto. Controlla se nella frase il verbo sia quello di interesse, abbia `POS = VERB` ed infine che sia legato al soggetto ed al complemento oggetto passato. In caso positivo ritorna `True`

- `semantic_type(sentence, word)`
  data una frase, ed una parola in input, esegue la WSD con `lesk` e poi ritorna il lexname del synset trovato. Se la parola è un pronome, viene subito ritornato il semantic type `'noun.person'`, senza fare wsd

- `extract_fillers(corpus)`
  metodo che prende in input il corpus, precedentemente parsificato tramite il metodo `parse_corpus()`, e restituisce tutte le coppie di filler legati al verbo in questione, con i relativi semantic types.

Risultati

Le coppie di tipi semantici sono state ordinate per frequenza di modo da evidenziare i comportamenti delle prime 15 più ricorrenti negli slot. Di seguito i risultati ottenuti




```
Frequenza Semantic Type del Soggetto: [('noun.person', 257), ('noun.group', 115), ('noun.cognition', 53),
('noun.act', 30), ('noun.artifact', 24), ('adj.all', 15), ('noun.location', 14), ('verb.social', 12),
('noun.communication', 11), ('noun.attribute', 10), ('verb.change', 10), ('noun.body', 10), ('verb.motion', 10),
('noun.event', 9), ('verb.creation', 8)]
```

```
Frequenza Semantic Type dell oggetto: [('noun.communication', 79), ('noun.cognition', 72), ('noun.act', 72),
('noun.artifact', 54), ('noun.group', 41), ('verb.motion', 32), ('noun.person', 31), ('verb.creation', 24),
('verb.communication', 20), ('noun.location', 20), ('verb.contact', 18), ('verb.social', 17), ('verb.change', 16),
('verb.competition', 15), ('noun.object', 14)]
```

Conclusioni

Questo esperimento ha evidenziato come sia difficile disambiguare un verbo per un calcolatore, vedendo la quantità di tipi diversi che sono stati individuati, ma anche come l'approccio di Patrick Hanks sia molto semplice da eseguire e produca risultati molto precisi.

Per il verbo create è più facile trovare una frase tipo, ad esempio: "*I created a chair*". Infatti, la coppia al primo posto è *"noun.person - noun.cognition"*

Alcuni accorgimenti

- *Word Sense Disambiguation*: per fare WSD e trovare il synset corretto è stata usata la funzione `lesk()` di wordnet che, come sappiamo, non funziona perfettamente e potrebbe restituire un synset non corretto

- *Individuazione supersensi*: è stata utilizzata la funzione `lexname()` per individuare i supersensi degli argomenti del verbo, è possibile che utilizzando altri approcci si ottengano risultati diversi (come CSI).