Esercitazione 1 - parte 1

La prima parte della prima esercitazione consiste nell'implementare alcune misure di similarità che sfruttano il grafo di wordnet, per valutare la similarità concettuale fra termini. Infine valutare i risultati utilizzando il file "WordSim353" fornito a lezione calcolando l'indici di correlazione di Spearman e di Pearson.

main.py

file da eseguire per la computazione. Inizialmente legge il file "WordSim353.tsv" e per ogni coppia di parole calcola le 3 misure di similarità implementate e descritte di seguito. Infine calcola la correlazione con l'annotazione presente nel file e stampa per ogni misura di similarità indici di correlazione di Spearman e Pearson.

similarity.py

file che contiene l'implementazione delle misure di similarità. Per ognuna di esse vi sono due metodi "_word" e "_sense" per un totale di 6. Questo perché le misure di similarità lavorano tra i sensi di wordnet, mentre il file contiene termini, quindi i metodi "_word" ciclano su tutte le possibili coppie di sensi delle due parole, calcolano la similarità utilizzando i metodi "_sense" e restituiscono il risultato massimo. I metodi "_sense" quindi implementano la misura vera e propria.

utils.py

Le misure di similarità utilizzano il grafo di wordnet per ottenere diversi valori. Questo file contiene i metodi per calcolare la profondità minima e massima di un nodo, il "Lowest Common Subsumer", ossia il primo antenato comune fra due nodi e il cammino più corto tra due nodi.

Wu & Palmer

spearman = 0.2791559033752252
pearson = 0.23133035067789123

$$\mathrm{cs}(s_1, s_2) = \frac{2 \cdot depth(\mathrm{LCS})}{depth(s_1) + depth(s_2)}$$

Shortest Path

spearman = 0.281253398737844
pearson = 0.18408849859573118

$$\mathrm{sim}_{path}(s_1, s_2) = 2 \cdot \mathrm{depthMax} - len(s_1, s_2)$$

Leakcock & Chodorow

spearman = 0.281253398737844
pearson = 0.31454299037672206

$$\mathrm{sim}_{LC}(s_1, s_2) = -\log \frac{len(s_1, s_2)}{2 \cdot \mathrm{depthMax}}$$

correlation.py

contiene l'implementazione delle misure di similarità. Viene utilizzata la libreria "pandas" per fare il rank delle variabili e numpy per calcolare la deviazione standard e la coovarianza.

Conclusioni

come possiamo vedere tutte le misure di similarità sono piuttosto basse, quindi queste misure sono ancora distanti dal ragionamento di un annotatore umano. Esempio di output:

```
['wup_spearman = 0.2791559033752252', 'sp_spearman = 0.281253398737844', 'lch_spearman = 0.281253398737844']
['wup_pearson = 0.23133035067789123', 'sp_pearson = 0.18408849859573118', 'lch_pearson = 0.31454299037672206']
```

Esercitazione 1 - parte 2

La seconda parte dell'esercitazione consiste nell'implementazione di un approccio bag of words (algoritmo di Lesk) per la disambiguazione di parole. In particolare si utilizza il corpus SemCor per calcolare l'accuratezza del sistema.

parte2.py

File da eseguire per avviare la computazione. L'algoritmo è testa su 10 esecuzioni di 50 frasi scelte casualmente dal corpus, per ognuna delle quali viene scelto un sostantivo causale da disambiguare. Infine stampa l'accuratezza del sistema. La funzione "calc_lesk" si occupa di randomizzare le frasi, scegliere il sostantivo e richiamare l'algoritmo di lesk vero e proprio.

disambiguation.py

Contiene l'implementazione dell'algoritmo vera e propria. La funzione lesk ha come input la parola da disambiguare e la frase in cui è contenuta e restituisce "None" se non vi sono sensi di wordnet per quella parola. Il contesto è quindi composto dalle parole della frase e la signature di ogni senso è composto dai termini della definizione e degli esempi relativi al senso in questione. Prima di calcolare l'overlap i termini del contesto vengono filtrati.

utils_es2.py

Contiene il metodo per filtrare i termini del contesto. Quindi elimina le stop_words contenute nel file fornito a lezione, e utilizza la funzione "morphy(word)" di wordnet per la lemmatizzazione. Infine elimina i duplicati e restituisce la lista.

Conclusioni

L'accuratezza media su 10 esecuzioni si aggira tra il 0,6 nei casi peggiori e 0,7 nei migliori. Un'esempio di esecuzione è il seguente:

```
0.68
0.66
0.74
0.72
0.64
0.68
0.7
0.64
0.74
0.66
tot = 0.686
```

Ogni riga corrisponde all'accuratezza su 50 frasi, e il totale all'accuratezza media dei valori precedenti.