**Università degli Studi di Torino**

Dipartimento di Informatica

Corsi di Laurea Magistrale in Informatica



**Relazione esercitazioni TLN**

**(prof. Radicioni)**

Salvatore Coluccia

a.a. 2019/2020

# Esercitazione 1 – Concept Similarity

Per questa esercitazione il file da lanciare è *similarity\_checker.py*.

Le tre misure di concept similarity implementate sono: WuPalmer, ShortestPath e LeakcockChodorow.

Per ognuna ho calcolato il coefficiente di similarità di Pearson e Spearman confrontando le similarità in output con quelle definite nel file WordSim353\WordSim353.csv

I risultati non evidenziano una forte similarità in quanto i coefficienti mi sembrano bassi ma in questo caso l’obiettivo principale non erano tanto le performance ma la sperimentazione in sé.

***WuPalmer\_pearson****: 0.25164180797402397*

***WuPalmer\_spearman****: 0.28340466601934766*

***ShortestPath\_pearson****: 0.1772591953361014*

***ShortestPath\_spearman****: 0.2935938806601189*

***LeakcockChodorow\_pearson****: -0.0677814736141495*

***LeakcockChodorow\_spearman****: 0.2935938806601189*

# Esercitazione 2 – Word sense disambiguation

Per questa esercitazione il file da lanciare è *word\_disambiguer.py*.

Ho implementato e usato l’algoritmo di Lesk e confrontato i risultati con un algoritmo dummy che sceglie sempre il primo senso del termine (il più frequente).

L’accuracy in output mi sembra soddisfacente:

**Accuracy**: 0.68

**Accuracy with dummy algo**: 0.84

Il dummy algorithm mi dà un'accuratezza più alta: Io individuerei come possibile causa il fatto che uso le words di tutti gli examples e quindi do maggiore rilevanza ai sense che hanno più esempi (o esempi più lunghi) in quanto hanno più probabilità di overlap con il contesto in input all’algoritmo di Lesk, questo potrebbe essere in effetti un bias da poter approfondire meglio.

# Esercitazione 3 – Framenet

Per questa esercitazione il file da lanciare è *framenet.py*.

Ho scelto di utilizzare il metodo bag of words per trovare il sysnset più simile per ogni elemento.

Il contesto di ogni synset di wordnet lo prendo da examples, hyponim, hyperonim e definition.

Creo il contesto di ogni frame usando le definition del frame e dei suoi frame elements (usando lemming e stop words elimination come preprocessing).

Le annotazioni manuali sono per i vari elementi sono individuabili come costanti all’interno del codice stesso (tutte le costanti che iniziano con **GOLD\_STANDARD\_\***

Queste sono le accuracy finali per ogni frame: 0.38 - 0.36 - 0.46 - 0.39 - 0.70

# Esercitazione 4 – Summarizer

Il file da lanciare per questa esercitazione è *summarizer.py*.

Nella versione finale ho utilizzato il file *dd-small-nasari-15* mettendolo in un dataframe pandas ed estraendo i termini dei Babelsynset. Ho effettuato anche delle prove utilizzando direttamente i web services di BabelNet (creandomi una api key) ma poi ho lasciato perdere per via del limite di chiamate giornaliero e per via del tempo di esecuzione molto più alto.

Ho utilizzato il metodo del Titolo per generare il contesto del documento sul quale comparare i contesti dei vari paragrafi.

Lo score di ogni paragrafo è dato dallo square-rooted Weighted Overlap, a tal proposito alcuni paragrafi in alcuni documenti presentano uno score uguale: probabilmente perché i termini del contesto di riferimento in quel caso non erano tanti e quindi sono sempre i termini più frequenti ad essere comuni.

# Esercitazione 5 – SemEval

Il file da lanciare per questa esercitazione è *semval\_lab.py* e i dati a supporto sono nella cartella *semval\_utils*.

In generale ho utilizzato la libreria pandas per la manipolazione dei dati tramite Dataframe in quanto mi ha permesso di effettuare calcoli veloci avendo una struttura facilmente accessibile.

Ho utilizzato il file SemEval17\_IT\_senses2synsets.txt per avere dei babel synset a partire da un termine.

Le coppie annotate manualmente per la consegna 1 sono nel file *semval\_utils\it.test.data.annotated.tsv*.

Di seguito i coefficienti di Spearman e Pearson per la consegna1: **0.475** e **0.474.**

Per la consegna 2 ho annotato le coppie di termini con i babelsynset che mi sembravano più appropriati: file *semval\_utils\it.test.data.annotated.consegna2.tsv*.

Per alcune coppie è capitato di non avere il babelsynset annotato a mano nel file *dd-nasari.txt*, in quei casi ho scartato la coppia dal processamento (si tratta comunque di un numero limitato rispetto al totale).

Ho quindi poi confrontato la mia annotazione con i synset trovati dall’algoritmo definito nella consegna: utilizzando il file *mini\_NASARI.tsv* ho calcolato la coppia di sensi che massimizzano lo score di similarità.

Di seguito le accuracy calcolate:

**Accuracy over first babel**: 0.5833333333333334

**Accuracy over second babel**: 0.4166666666666667

**Accuracy over couples**: 0.2916666666666667