Dati in input due termini, il task di conceptual similarity consiste nel fornire uno score di similarità che ne indichi la vicinanza semantica. Per esempio, la similarità fra i concetti *car* e *bus* potrebbe essere 0.8 in una scala [0,1], in cui 0 significa che i sensi sono completamente dissimili, mentre 1 significa identità.
Per risolvere il task di conceptual similarity è possibile sfruttare la struttura ad albero di WordNet.
L'input per questa esercitazione è costituito da coppie di termini contenute nel file *WordSim353*. Il file contiene 353 coppie di termini. A ciascuna coppia è attribuito un valore numerico [0,10], che rappresenta la similarità fra gli elementi della coppia.
L'esercitazione consiste nell'implementare tre misure di similarità basate su WordNet :

- **Wu & Palmer**
  Questa metrica si basa sulla struttura di WordNet e la similarità tra due synset si calcola come

$$cs(s1, s2) = \frac{2 * depth(LCS)}{depth(s1) + depth(s2)}$$

  Dove LCS è il più basso antenato comune (Lowest Common Subsumer) fra i synset $s1$ e $s2$ e $depth(x)$ è la funzione che misura la distanza fra la radice di WordNet e il synset x.

- **Shortest Path**

$$sim_{path} = 2 * depthMax - len(s1, s2)$$

  Dove *DepthMax* è la profondità massima di WordNet (valore fisso) e $len(s1, s2)$è la lunghezza del percorso più breve che collega i due synset $s1$ e $s2$.

  La similarità tra i due sensi $s1$ e $s2$ è funzione della lunghezza del percorso più breve che collega i due synset:
  - Se $len(s1, s2) = 0$ allora il valore di similarità assume è massimo, ovvero $2 \cdot depthMax$;
  - Se $len(s1, s2) = 2 \cdot depthMax$ allora la similarità è minima e quindi uguale a 0

- **Leakcock and Chodorow**

$$sim_{LC}\ (s1, s2) = -\log\ (\ len(s1, s2)/(2 * depthMax))$$

Quando $s1$ e $s2$ hanno lo stesso senso, $len(s1, s2) = 0$, quindi per evitare $\log(0)$ si aggiunge 1 sia al numeratore che al denominatore. Quindi il valore di similarità è compreso nell'intervallo $[0, \log(2 \cdot depthMax + 1)\ ]$

Attenzione: l'input è costituito da coppie di termini, mentre la formula utilizza sensi.
Per calcolare la similarità fra due termini immaginiamo di prendere la massima similarity fra tutti i sensi del primo termine e tutti i sensi del secondo termine.
Quindi i due termini funzionano come contesto di disambiguazione l'uno per l'altro. Nella formula *c* sono i concetti che appartengono ai synset associati ai termini w1 e w2.La massima similarità tra due termini si calcola come:

$$sim(w1, w2) = max_{c1 \in s(w1),\ c2 \in s(w2)}[sim(c1, c2)]$$

Per ciascuna delle misure di similarità ricavate, si calcolano gli indici di correlazione di Spearman e gli indici di correlazione di Pearson le suddette misure e quelle *target* presenti nel file annotato.

- *Indice di correlazione di Pearson*

Assegnate le serie di dati $A = \{x_1, \ldots, x_n\}$ e $B = \{y_1, \ldots, y_n\}$ si definisce **coefficiente di correlazione** campionario, o **indice di correlazione** di Pearson, il seguente valore numerico

$$\rho_{AB} = \frac{\operatorname{cov}(A, B)}{\sigma_A \cdot \sigma_B}$$

dove $\operatorname{cov}(A, B)$ indica la covarianza di A e B. σ_A ed σ_B indicano, rispettivamente, la deviazione standard campionaria di A e B.

- *Indice di correlazione di Spearman*

L'indice di correlazione R per ranghi di Spearman è una misura statistica non parametrica di correlazione. Essa misura il grado di relazione tra due variabili e l'unica ipotesi richiesta è che siano ordinabili, e, se possibile, continue.

A livello pratico il coefficiente ρ è semplicemente un caso particolare del coefficiente di correlazione di Pearson dove i valori vengono convertiti in ranghi prima di calcolare il coefficiente.

$$\rho_{rgX,\ rgY} = \frac{cov(rgX,\ rgY)}{\sigma(rgX, rgY)}$$

1.1 Svolgimento

Dopo aver creato le liste `wup_distance, sp_distance e lc_distance`, che conterranno i risultati, per ogni coppia di parole di parole presenti nel file annotato, si determinano i synset associati ad entrambi i termini tramite la funzione NLTK `wn.synsets(word)`. I synset vengono poi utilizzati per calcolare tre score di similarità tra le parole, uno per ogni metrica. I vari score vengono via via aggiunti alle liste precedentemente create.

Per poter utilizzare le formule relative alle tre metriche da calcolare è stato necessario implementare dei metodi particolari e calcolare quindi gli elementi *LCS* e *len(s1, s2):*

- `commonHypernyms()`
- `getLowestCommonHypernym()`
- `minimumDistanceImproved()`

La funzione `commonHypernyms()` permette di determinare la lista degli iperonimi comuni ai due synset in input. Inizialmente vengono calcolati tutti gli iperonimi dei due synset, tramite il metodo implementato in NLTK `synset.hypernym_paths()`. Le liste di iperonimi vengono intersecate per otttenere gli iperonimi in comune.
La funzione `getLowestCommonHypernym()` determina, in seguito, l'antenato comune che è più vicino ai synset in input (quello più lontano dal root), ricavato utilizzando la funzione `maxDepth(listOfHypernyms)`, sulla lista di iperonimi precedentemente trovata tramite il metodo `commonHypernyms()`.

La funzione `minimumDistanceImproved()` calcola la lunghezza del percorso più breve che collega due synset. È pensata sul modello del BFS algorithm, infatti viene utilizzato un loop che va a percorrere l'albero a partire dai due synset in input, fino alla radice, attraverso la funzione di NLTK hypernyms(synset) (che restituisce gli iperonimi ,quasi sempre 1, del synset) finchè non sono uguali, aggiornando di volta in volta il valore distance. Nel caso in cui siano uguali distance è uguale a 0, nel caso peggiore è uguale all'altezza dell'albero dalla radice fino al nodo da cui siamo partiti. Il calcolo della metrica viene effettuato calcolando prima tutte le distanze tra i vari sensi delle due parole in input, che vengono salvate in una lista e poi cercando il minimo valore che verrà usato nella formula (*len(s1,s2)).*
Il valore della profondità massima di WordNet, necessaria per il calcolo della similarità con Shortest Path e Leakcock & Chodorow, corrisponde a 20 e viene calcolata come:

```
max(max(len(hyp_path) for hyp_path in ss.hypernym_paths()) for ss in wn.all_synsets())
```

Dopo aver calcolato la similarità tra i termini è necessario calcolare la correlazione tra le similarità calcolate e quelle target (presenti nel file in input *WordSim353.csv*). Siccome le similarità target e quelle relative alle metriche Shortest Path e Leakcock & Chodorow hanno un range di valori diverso vengono normalizzate.
Gli indici di correlazione vengono calcolate tramite il coefficiente di Pearson e quello di Spearman e il risultato è il seguente:

Correlazione tra valori di similarità calcolati e quelli target

Similarity metric	Spearman index	Pearson index
Wu & Palmer	0.291	0.322
Shortest Path	0.136	0.206
Leakcock & Chodorow	0.233	0.206