

UNIVERSITA’ DEGLI STUDI DI TORINO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Anno Accademico 2024/2025

**Tecnologie del Linguaggio Naturale-Parte II Prof. D. Radicioni**

Autore: Annalisa Sabatelli Matr. 866879

ESERCITAZIONE 1

Word Sense Disambiguation (WSD)

1. **Introduzione**

L’obiettivo dell’esercitazione WSD è quello di implementare una versione dell'algoritmo di Lesk al fine di permettere la disambiguazione del significato delle parole.

Dal corpus annotato SemCor vengono selezionate casualmente 50 frasi. Per ciascuna frase, si identifica un sostantivo che soddisfa specifici criteri (definiti in seguito), su cui viene applicato l’algoritmo di disambiguazione.

Il processo viene ripetuto per un numero prefissato di iterazioni (n\_iterazioni = 10).

Al termine di questo, viene calcolata l’accuratezza media dell’algoritmo nel determinare correttamente il senso delle parole polisemiche. Tale valore viene poi confrontato con l’accuratezza ottenuta utilizzando l’implementazione dell’algoritmo di Lesk disponibile nella libreria NLTK.

Per la disambiguazione è stato adottato un approccio basato sul modello Bag of Words, in cui le parole del contesto vengono confrontate con i termini presenti nelle definizioni dei sensi associati al lemma.

1. **Struttura del codice**

Il codice è diviso in 3 file:

• *utils.py*: contiene metodi di utility tra cui

* *def extraction\_lemmi\_from\_sentence(sentence):* metodo che prende in input una frase e la elabora restituendo un set costituito dai lemmi delle parole che la costituiscono. In particolare, l’elaborazione si articola nei seguenti passi:
  + rappresentazione di tutte le parole in lower case;
  + rimozione delle stop-words;
  + rimozione della punteggiatura;
  + rimozione degli spazi vuoti;
  + creazione di una lista di lemmi;
  + trasformazione della lista in un set.
* *def extraction\_terms\_from\_corpus(num\_words):* metodo che restituisce una lista di cardinalità *num\_words* di termini ambigui in WordNet e presenti nel corpus Semcor.

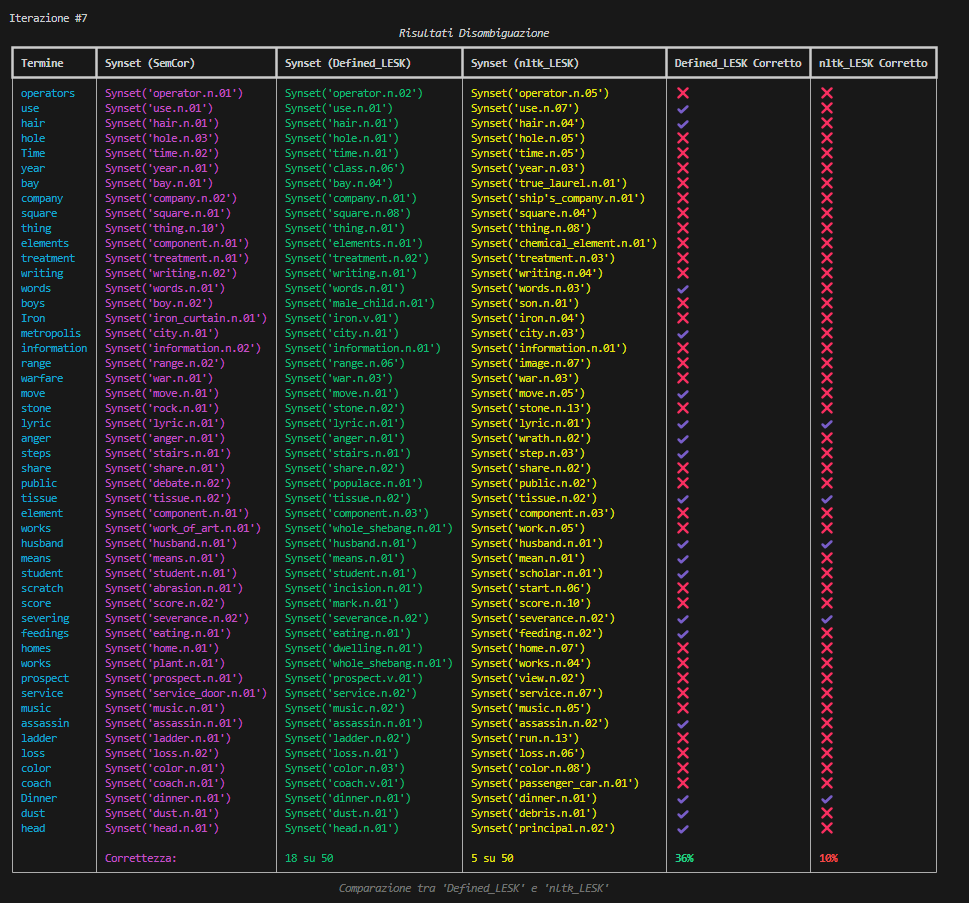
• *LeskAlgorithm.py*: contiene il metodo che implementa l’algoritmo di Lesk senza l’utilizzo della libreria NLTK. In particolare:

* *def lesk\_algorithm(term, sentence)*: metodo che prende in input la parola da disambiguare e la frase che contiene la parola e successivamente, per ogni synset della parola, tramite un approccio bag of word, calcola la misura di overlap tra i termini nella frase passata in input e i termini presenti nelle informazioni del synset trattato. Al termine ritorna il synset con massimo overlap.

• *main.py:* contiene il metodo main*.* Dopo aver verificato e scaricato le risorse necessarie da NLTK, come WordNet e SemCor, il programma esegue l’esperimento per un numero predefinito di iterazioni (10). In ogni iterazione, vengono selezionati casualmente 5 termini (sostantivi) dal corpus SemCor tramite la funzione di utilità descritta al punto precedente. Successivamente, viene calcolata la disambiguazione sia per la versione personalizzata dell’algoritmo di Lesk che per quella di NLTK. Alla fine delle iterazioni, viene calcolata l'accuratezza media di entrambi gli algoritmi e stampata in percentuale. Questo approccio consente un confronto diretto tra le due implementazioni in termini di prestazioni*.*

1. **Risultati ottenuti**

Si riporta, di seguito a titolo di esempio, i risultati ottenuti a valle di una delle 10 iterazioni eseguite.



Accuratezza media 'coded\_LESK': **45.6%**

Accuratezza media 'nltk\_LESK': **23,2%**

1. **Conclusioni**

L'accuratezza media ottenuta sulle dieci esecuzioni si attesta intorno al 45,6%. Tale risultato può essere attribuito, da un lato, alla limitata lunghezza di alcune frasi presenti nel corpus che, combinata con l’approccio BOW (Bag of Words), non fornisce un contesto sufficiente per una distinzione efficace dei diversi sensi. Dall’altro lato, si osservano leggere discrepanze tra i POS tag presenti nel corpus e quelli assegnati dallo strumento NLTK: in alcuni casi, termini etichettati come NN nel corpus non vengono riconosciuti come tali da NLTK, ma classificati diversamente (ad esempio come ADJ), influenzando negativamente le prestazioni complessive del sistema.