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.

2. 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.

3. Risultati ottenuti

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

Iterazione #7

Risultati Disambiguazione

Termine	Synset (SemCor)	Synset (Defined_LESK)	Synset (nltk_LESK)	Defined_LESK Corretto	nltk_LESK Corretto
operators	Synset('operator.n.01')	Synset('operator.n.02')	Synset('operator.n.05')	✗	✗
use	Synset('use.n.01')	Synset('use.n.01')	Synset('use.n.07')	✓	✗
hair	Synset('hair.n.01')	Synset('hair.n.01')	Synset('hair.n.04')	✓	✗
hole	Synset('hole.n.03')	Synset('hole.n.01')	Synset('hole.n.05')	✗	✗
Time	Synset('time.n.02')	Synset('time.n.01')	Synset('time.n.05')	✗	✗
year	Synset('year.n.01')	Synset('class.n.06')	Synset('year.n.03')	✗	✗
bay	Synset('bay.n.01')	Synset('bay.n.04')	Synset('true_laurel.n.01')	✗	✗
company	Synset('company.n.02')	Synset('company.n.01')	Synset('ship's_company.n.01')	✗	✗
square	Synset('square.n.01')	Synset('square.n.08')	Synset('square.n.04')	✗	✗
thing	Synset('thing.n.10')	Synset('thing.n.01')	Synset('thing.n.08')	✗	✗
elements	Synset('component.n.01')	Synset('elements.n.01')	Synset('chemical_element.n.01')	✗	✗
treatment	Synset('treatment.n.01')	Synset('treatment.n.02')	Synset('treatment.n.03')	✗	✗
writing	Synset('writing.n.02')	Synset('writing.n.01')	Synset('writing.n.04')	✗	✗
words	Synset('words.n.01')	Synset('words.n.01')	Synset('words.n.03')	✓	✗
boys	Synset('boy.n.02')	Synset('male_child.n.01')	Synset('son.n.01')	✗	✗
Iron	Synset('iron_curtain.n.01')	Synset('iron.v.01')	Synset('iron.n.04')	✗	✗
metropolis	Synset('city.n.01')	Synset('city.n.01')	Synset('city.n.03')	✓	✗
information	Synset('information.n.02')	Synset('information.n.01')	Synset('information.n.01')	✗	✗
range	Synset('range.n.02')	Synset('range.n.06')	Synset('image.n.07')	✗	✗
warfare	Synset('war.n.01')	Synset('war.n.03')	Synset('war.n.03')	✗	✗
move	Synset('move.n.01')	Synset('move.n.01')	Synset('move.n.05')	✓	✗
stone	Synset('rock.n.01')	Synset('stone.n.02')	Synset('stone.n.13')	✗	✗
lyric	Synset('lyric.n.01')	Synset('lyric.n.01')	Synset('lyric.n.01')	✓	✓
anger	Synset('anger.n.01')	Synset('anger.n.01')	Synset('wrath.n.02')	✓	✗
steps	Synset('stairs.n.01')	Synset('stairs.n.01')	Synset('step.n.03')	✓	✗
share	Synset('share.n.01')	Synset('share.n.02')	Synset('share.n.02')	✗	✗
public	Synset('debate.n.02')	Synset('populace.n.01')	Synset('public.n.02')	✗	✗
tissue	Synset('tissue.n.02')	Synset('tissue.n.02')	Synset('tissue.n.02')	✓	✓
element	Synset('component.n.01')	Synset('component.n.03')	Synset('component.n.03')	✗	✗
works	Synset('work_of_art.n.01')	Synset('whole_shebang.n.01')	Synset('work.n.05')	✗	✗
husband	Synset('husband.n.01')	Synset('husband.n.01')	Synset('husband.n.01')	✓	✓
means	Synset('means.n.01')	Synset('means.n.01')	Synset('mean.n.01')	✓	✗
student	Synset('student.n.01')	Synset('student.n.01')	Synset('scholar.n.01')	✓	✗
scratch	Synset('abrasion.n.01')	Synset('incision.n.01')	Synset('start.n.06')	✗	✗
score	Synset('score.n.02')	Synset('mark.n.01')	Synset('score.n.10')	✗	✗
severing	Synset('severance.n.02')	Synset('severance.n.02')	Synset('severance.n.02')	✓	✓
feedings	Synset('eating.n.01')	Synset('eating.n.01')	Synset('feeding.n.02')	✓	✗
homes	Synset('home.n.01')	Synset('dwelling.n.01')	Synset('home.n.07')	✗	✗
works	Synset('plant.n.01')	Synset('whole_shebang.n.01')	Synset('works.n.04')	✗	✗
prospect	Synset('prospect.n.01')	Synset('prospect.v.01')	Synset('view.n.02')	✗	✗
service	Synset('service_door.n.01')	Synset('service.n.02')	Synset('service.n.07')	✗	✗
music	Synset('music.n.01')	Synset('music.n.02')	Synset('music.n.05')	✗	✗
assassin	Synset('assassin.n.01')	Synset('assassin.n.01')	Synset('assassin.n.02')	✓	✗
ladder	Synset('ladder.n.01')	Synset('ladder.n.02')	Synset('run.n.13')	✗	✗
loss	Synset('loss.n.02')	Synset('loss.n.01')	Synset('loss.n.06')	✗	✗
color	Synset('color.n.01')	Synset('color.n.03')	Synset('color.n.08')	✗	✗
coach	Synset('coach.n.01')	Synset('coach.v.01')	Synset('passenger_car.n.01')	✗	✗
Dinner	Synset('dinner.n.01')	Synset('dinner.n.01')	Synset('dinner.n.01')	✓	✓
dust	Synset('dust.n.01')	Synset('dust.n.01')	Synset('debris.n.01')	✓	✗
head	Synset('head.n.01')	Synset('head.n.01')	Synset('principal.n.02')	✓	✗
	Correttezza:	18 su 50	5 su 50	36%	10%

Comparazione tra 'Defined_LESK' e 'nltk_LESK'

Accuratezza media 'coded_LESK': **45.6%**

Accuratezza media 'nltk_LESK': **23,2%**

4. 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.