Università degli studi di Torino

Dipartimento di informatica

Tecnologie del Linguaggio Naturale

Parte 2

Author
Giuseppe Biondi

21 ottobre 2021

Indice

1 Esercizio 1

1.1 Similarità

In questo esercizio importo le parole contenute nel file *WordSim353.csv* e calcolo le similarità: **Wu and Palmer, Shortest path, Leakcock and Chodorow**.

Per quanto riguarda la similarità di Wu and Palmer son state create due implementazioni: nella prima si forza il passaggio dal *subsumer* comune durante il calcolo della profondità. La seconda al contrario applica la formula senza nessun controllo e di conseguenza il calcolo di *depth(syn)* può considerare un percorso che evita il subsumer comune.

Riguardo il metodo *lowes-common-subsumer* l'interpretazione data è quella di cercare l'antenato comune meno distante dai synset in input

1.2 Word Sense Disambiguation

L'esercizio di disambiguazione sfrutta il dataset di brown *br-a01.xml*. L'idea è di estrarre tutte le frasi e iterare k volte usandone una a caso. Per ogni frase si sceglie una parola casuale al suo interno e si applica l'algoritmo di Lesk semplificato.

Per determinare il synset corretto in wordnet, si effettua una ricerca sul lemma della parola concatenato all'attributo lexsn (riga 125). Tuttavia può capitare che questa ricerca non trovi alcun synset probabilmente per problemi di compatibilità tra dataset e libreria nltk. In tal caso lo si considera come errore e lo si esclude dal calcolo della precisione.

Raramente possono capitare casi in cui la ricerca della parola su WordNet non ottiene alcun risultato, come per esempio per via della presenza dell'underscore nella parola stessa, in tal caso si considera come un risultato sbagliato.

Il comando a riga 148 *explaination-on* permette di stampare le liste di parole degli overlap per avere più dettagli su ogni itarazione

2 Esercizio 2

2.1 Framenet

Questo esercizio consiste nell'assegnare automaticamente dei synset di WordNet alle parti dei frame in FrameNet.

Per valutare il risultato è stato creato manualmente un file atteso: *valutation_file.txt*.

Alcuni termini non permettono di ottenere risultati nella ricerca su WordNet (i.e., *Transfer_scenario*) si è deciso allora di aggiungere tra parentesi quadre un termine più generale da usare per la ricerca del synset (i.e., *Trans-*

fer_scenario[Transfer] riga 46 del *valutation_file.txt*).

Son state implementati due criteri per la ricerca del synset migliore, entrambi si basano sul massimizzare uno score, rispettivamente **bag of words score** e **graph score**.

A riga 95 impostando la variabile flag a 1 si imposta l'esecuzione usando il graph score, rimuovendola o settandola a 0 si userebbe il bag of words score.

Dal momento che entrambi i criteri di score sfruttano il concetto di bag of word su framenet, si è potuto sperimentare due tipi di bag of words-context, la prima viene creata usando la definizione del singolo elemento su cui stiamo cercando il synset(i.e., un frame element), la seconda considera l'intero frame, cioè tutte le definizioni di tutti gli elementi del frame.

Nel file *log.txt* son portati dei risultati ottenuti al variare del tipo di bag of word passata e del tipo di score scelto. Si è provato anche ad aumentare il valore l a riga 46, impostato a 3, ma non da risultati molto differenti. Nel caso della bag of words più grande, sull'intero frame, aumentandolo richiedeva un'enorme quantità di tempo per completare la ricerca.

3 Esercizio 3

3.1 Summarization

Il seguente esercizio ha lo scopo di creare un riassunto dei testi contenuti nel path *'util/docs/'* applicando una riduzione del 10/20/30% in base al valore impostato nella variabile *_cut_ratio* a riga 128. In fine sarà applicata una valutazione confrontando i risultati con il riassunto automatico ottenuto con le librerie di *gensim*.

Per effettuare il riassunto si parte dall'estrazione del topic: si scelgono come termini di interesse quelli presenti nel titolo e le 10 parole più frequenti nel testo, a seguito di pre-processing.

Da queste parole, grazie alla risorsa di *Babelfy* si ottengono i BabelNet Synset già disambiguati e per ognuno si cercano i vettori di Nasari.

In seguito si crea una lista di vettori Nasari per il contesto: per ogni vettore ottenuto dal topic si estraggono i lemmi e si cercano i vettori corrispondenti usando lo stesso processo di disambiguazione tramite *Babelfy*.

In fine si crea lo scoring dei paragrafi andando a calcolare la similarità sia sul topic che sul contesto e pesando la metà quest'ultima in quanto potrebbe essere meno informativa.

In base al valore di *_cut_ratio* si selezionano solo i paragrafi da tenere e si creano i file di risultato nella cartella *docs/result/*: il file *difference.txt* contiene

le frasi rimosse nella creazione di *my result* e le frasi rimosse dalla creazione di *gold*, cioè il nostro risultato atteso.

4 Esercizio 4

4.1 Semantic Similarity

Per questo esercizio è stato annotato manualmente il file */utils/it.test.data.expected.tsv* con le coppie di parole e il valore di similarità atteso nel range 0-4.

Tali valori saranno confrontati con i risultati ottenuti dal programma.

Il processo inizia con la creazione delle risorse di Nasari e BabelNet, per quest'ultima si crea una *Dictionary* contenente come chiave la parola e come valore la lista di BabelNet id estratti con essa.

In seguito per ogni coppia di parole di cui bisogna calcolare la similarità, si prendono i rispettivi Synset e per ogni combinazione tra le due liste si calcola la cosine similarity sui vettori di Nasari. Il risultato finale sarà la similarità massima tra quelle ottenute.

Nel caso in cui ci fosse qualche BabelNet id il cui vettore di Nasari non è presente nella nostra risorsa si salterebbe al confronto successivo.

I risultati ottenuti vengono stampati nel file *utils/result_valutation.tsv*: parola 1, parola2, risultato atteso, risultato ottenuto.

4.2 Sense Identification

In questo caso il programma cercherà di trovare i Synset pensati dall'utente nella fase di annotazione.

Come per l'esercizio precedente si inizia con l'annotazione manuale del file: */utils/si.test.data.expected.tsv*. Segue l'inizializzazione delle risorse BabelNet e Nasari.

Il processo è molto simile, l'unica differenza è che in questo caso si estraggono i Synset che massimizzano la cosine similarity e di conseguenza il file risultante *utils/result_identification.tsv* conterrà due righe per ogni coppia di parole: la prima con i risultati attesi e la seconda con quelli predetti.