**Università degli Studi di Torino**

Dipartimento di Informatica

Corsi di Laurea Magistrale in Informatica



**Relazione esercitazioni TLN**

**(prof. Di Caro)**

Salvatore Coluccia

a.a. 2019/2020

## Esercitazione 1

Questa esercitazione prevede il calcolo di similarità tra le definizioni date per 4 concetti mostrati a lezione.

L’obiettivo del laboratorio è calcolare in qualche modo l’agreement nelle definizioni degli studenti e capire se ci sono concetti per i quali si nota una netta differenza nelle definizioni.

L’algoritmo che ho utilizzato per calcolare queste percentuali è molto semplice e si divide in due macro-step:

1. Lemmatizzazione e rimozione delle stopword
2. Calcolo similarità tra tutte le definizioni di un dato concetto

Per la fase 1 ho utilizzato le funzioni di nltk che mette già a disposizione un elenco di stopwords inglesi che ho semplicemente rimosso dalle frasi che sono state precedentemente tokenizzate sempre utilizzando nltk.  
Sempre tramite la stessa libreria ho potuto procedere alla lemmatizzazione dei token trovati.

Per la fase 2 ho calcolato l’overlap medio dei lemmi delle definizioni di ogni concetto.

I risultati ottenuti sono i seguenti:

|  |  |  |
| --- | --- | --- |
|  | ASTRATTO | CONCRETO |
| GENERICO | 8.2% (Freedom) | 24.6% (Building) |
| SPECIFICO | 8.47% (Compassion) | 15.8% (Molecule) |

Dai risultati, per quanto chiaramente non possano essere dei dati scientificamente rilevanti dato il campione molto piccolo di esempi, si evince un agreement nettamente superiore per i concetti di tipo Concreto.  
Intuitivamente era in effetti quello che mi aspettavo in quanto descrivere un concetto concreto risulta generalmente più semplice di un concetto più astratto.

## Esercitazione 2

In questa esercitazione l’obiettivo è quello di identificare il concetto a partire da un insieme di definizioni.

Gli step che ho seguito per completare questo risultato sono i seguenti (ripetuti per ogni concetto):

1. Identificazione dei token, lemmatizzazione e rimozione stopwords da ogni definizione
2. Calcolo frequenza token considerando tutte le definizioni
3. Trovo contesto utilizzando examples,definitions, iperonimi e iponimi di ogni synset associato ad un termine (il synset associato ad ogni termine lo trovo applicando l’algoritmo di Lesk e usando i token trovati nelle definizioni come contesto)
4. Genero una lista con tutti gli iponimi di ogni iperonimo di ogni synset. Durante la creazione di questa lista considero iperonimi i termini che hanno una frequenza maggiore di una certa soglia (e quindi per questi conduco la ricerca sui loro iponimi)
5. Per ogni concetto ritorno l’iponimo che è più simile (in termini di overlap del suo contesto col contesto trovato al punto 3).

Questi sono i risultati ottenuti:

|  |  |
| --- | --- |
| CORRETTO | SYNSET IDENTIFICATO |
| JUSTICE | human\_right.n.01 |
| PATIENCE | excitement.n.02 |
| GREED | one.n.02 |
| POLITICS | government.n.03 |
| FOOD | plant.n.04 |
| RADIATOR | dispersion.n.03 |
| VEHICLE | airlift.n.01 |
| SCREW | pin.n.09 |

## Esercitazione 3

In questa esercitazione si vogliono sperimentare i principi della teoria di Hanks e quindi si cerca di calcolare delle frequenze di supersensi WordNet dato un corpus e scelto un verbo e una valenza.

Io ho scelto una valenza 2, usato il Brown Corpus e scelto il verbo build.

L’algoritmo esegue i seguenti macro step:

1. Estrae dal brown corpus le frasi che contengono il verbo TO BUILD
2. Da queste frasi estrae una lista di coppie che rappresentano i supersensi WordNet associati ai termini rispettivamente a sinistra e a destra del verbo in questione
3. Infine aggrega i risultati raggruppando per coppie uguali e visualizzandone la relativa frequenza

Questi sono i risultati ottenuti:

|  |  |
| --- | --- |
| noun.possession\_\_\_adj.all: 2  verb.possession\_\_\_verb.change: 1  verb.stative\_\_\_adj.all: 2  noun.group\_\_\_noun.artifact: 1  noun.group\_\_\_noun.attribute: 1  noun.group\_\_\_noun.group: 2  adj.all\_\_\_noun.artifact: 1  verb.emotion\_\_\_noun.time: 2  adv.all\_\_\_adj.pert: 1  noun.communication\_\_\_noun.attribute: 1  verb.stative\_\_\_verb.competition: 1  noun.group\_\_\_adj.all: 1  verb.possession\_\_\_noun.phenomenon: 1  noun.possession\_\_\_noun.group: 1  verb.contact\_\_\_noun.phenomenon: 1  noun.process\_\_\_adj.all: 1  noun.person\_\_\_adj.all: 1  noun.object\_\_\_noun.relation: 1  noun.attribute\_\_\_adj.all: 1  noun.act\_\_\_noun.attribute: 2  noun.body\_\_\_adj.all: 1  verb.cognition\_\_\_noun.artifact: 2  noun.time\_\_\_noun.artifact: 1  noun.artifact\_\_\_noun.state: 1  adj.all\_\_\_adj.all: 3  noun.substance\_\_\_adj.all: 1  noun.state\_\_\_noun.person: 1  noun.state\_\_\_noun.artifact: 1 | noun.state\_\_\_noun.cognition: 1  verb.possession\_\_\_adj.all: 1  verb.social\_\_\_noun.person: 1  adv.all\_\_\_adj.all: 1  adj.all\_\_\_noun.group: 1  noun.state\_\_\_noun.state: 1  verb.cognition\_\_\_noun.quantity: 1  noun.person\_\_\_noun.artifact: 2  noun.object\_\_\_adj.all: 1  noun.person\_\_\_noun.communication: 1  noun.state\_\_\_noun.quantity: 1  noun.time\_\_\_noun.relation: 1  noun.substance\_\_\_adj.pert: 1  noun.substance\_\_\_verb.social: 1  noun.artifact\_\_\_adv.all: 1  noun.state\_\_\_noun.attribute: 1  adv.all\_\_\_verb.contact: 1  noun.person\_\_\_verb.stative: 1  adj.all\_\_\_adv.all: 1  noun.location\_\_\_verb.stative: 1  noun.substance\_\_\_noun.time: 1  noun.attribute\_\_\_noun.process: 1  adv.all\_\_\_noun.quantity: 1  noun.event\_\_\_noun.artifact: 1  noun.group\_\_\_noun.location: 1  noun.act\_\_\_noun.artifact: 1  adj.all\_\_\_verb.communication: 1  noun.substance\_\_\_noun.artifact: 1  verb.change\_\_\_noun.substance: 1  adv.all\_\_\_noun.cognition: 1  adj.all\_\_\_noun.person: 1  noun.state\_\_\_adv.all: 1  verb.perception\_\_\_noun.quantity: 1  noun.event\_\_\_verb.cognition: 1  noun.time\_\_\_noun.cognition: 1  adv.all\_\_\_noun.phenomenon: 1  noun.artifact\_\_\_noun.artifact: 1 |

## Esercitazione 4

In questa esercitazione l’obiettivo è fornire un’implementazione di un algoritmo di segmentazione valido per un testo qualsiasi in modo da individuare i “cambi di contesto” in modo soddisfacente.

Il mio algoritmo esegue i seguenti step (sull’approccio text tiling):

1. Trovo le frasi e i relativi token
2. Lemming ed eliminazione stop words
3. Suddivido le frasi in un numero di finestre iniziali predefinito
4. Per ogni finestra calcolo la coesione media delle coppie di frasi in modo sequenziale utilizzando la wuPalmer similarity tra i synset wordnet associati ai token delle frasi e prendendo per ogni coppia di termini la max(similarity)
5. Se la coesione media è più bassa di una certa soglia allora divido la window in due sotto windows sulla base del break point individuato (la coppia di frasi meno simili). Inoltre diminuisco la soglia minima di un fattore pari a 0.7 per evitare “overfitting” e ridurre al minimo le window con solo una frase
6. Ripeto gli step 4 e 5 fino a quando non ci sono più split oppure arrivo al numero massimo di iterazioni

I risultati ottenuti possono essere considerati un primo risultato che sicuramente è migliorabile ma che comunque riesce a suddividere in modo un po’ grossolano i vari contesti.

Le performance migliori sono state ottenute con un numero iniziale di finestre pari a 5 e con una soglia minima di coesione per una window pari a 0.6.

Il testo utilizzato per i test è stato creato a partire da diversi testi relativi a diverse news in modo da poter debuggare con meno ambiguità l’algoritmo.