Università degli Studi di Torino

Corso di Laurea Magistrale in Informatica

Tecnologie del Linguaggio Naturale

Docente: Luigi Di Caro

**Relazione esercitazioni**

Alessandro Clocchiatti

Matricola 909105

Questa relazione riguarda le esercitazioni della terza parte del corso e comprende:

1. Definition similarity
2. Content-to-form
3. Semantic clustering (Hanks)
4. Text segmentation
5. OIE system

**1. Esercitazione: Definition similarity**

**1.1 Consegna**

L’esercitazione prevede i seguenti passaggi:

1. Caricamento dei dati sulle definizioni (file *definizioni.xls* o documento Google presente su Moodle);
2. Preprocessing (su frequenza minima dei termini, stemming, etc. a vostra scelta);
3. Calcolo similarità tra definizioni (cardinalità dell’intersezione dei termini normalizzata su lunghezza minima tra le due, o varianti a scelta);
4. Aggregazione sulle due dimensioni (concretezza / specificità come da schema in basso);
5. Interpretazione dei risultati e scrittura di un piccolo report (da inserire nel vostro portfolio per l’esame).

|  |  |  |
| --- | --- | --- |
|  | **Astratto** | **Concreto** |
| **Generico** | % | % |
| **Specifico** | % | % |

**1.2 Svolgimento**

Le definizioni presenti nel file *definizioni.csv* sono state fornite da diverse persone e riguardano i seguenti termini:

* building (concreto generico);
* molecule (concreto specifico);
* freedom (astratto generico);
* compassion (astratto specifico).

Dopo aver letto il file, viene applicato un pre-processing ad ogni definizione, attraverso le seguenti operazioni:

* tokenizzazione;
* rimozione stopword;
* rimozione punteggiatura;
* stemming.

Attraverso la funzione getSimilarity() si calcola la similarità tra le definizioni pre-processate riferite allo stesso termine. La funzione calcola, per ogni possibile coppia di definizioni, il numero di token comuni alle due definizioni. Il token si riferisce ad un termine/elemento della definizione.

La similarità tra due definizioni si calcola come

Ovvero la cardinalità dell'insieme intersezione tra gli insiemi normalizzata sulla cardinalità minima dei due insiemi.

La similarità non viene calcolata nel caso in cui la coppia sia formata dallo stesso elemento.

Infine la similarità globale tra le definizioni riferite ad un termine si ottiene attraverso la media delle similarità tra coppie.

Il risultato ottenuto è il seguente:

|  |  |  |
| --- | --- | --- |
|  | **Astratto** | **Concreto** |
| **Generico** | 9% | 26% |
| **Specifico** | 14% | 16% |

Si può notare che le definizioni riguardanti i termini astratti abbiano meno termini in comune rispetto a quelli concreti. Inoltre, mentre le definizioni del concetto concreto generico risultano più simili tra loro rispetto a quelle del termine specifico concreto (10% più simili), nel caso dei termini astratti avviene il contrario (le definizioni del concetto astratto specifico risultano più simili rispetto a quelle del termine astratto generico).

**2. Esercitazione: Definition similarity**

**2.1 Consegna**

L’esercitazione prevede i seguenti passaggi:

1. Caricamento dei dati content-to-form (presente su Moodle);
2. Preprocessing (si veda esercitazione precedente, a vostra scelta);
3. Utilizzo di WordNet come sense inventory, per inferire il concetto descritto dalle diverse definizioni;
4. Definire ed implementare un algoritmo (efficace ma anche efficiente) di esplorazione dei sensi di WordNet, usando concetti di similarità (tra gloss e definizioni, esempi d’uso, rappresentazioni vettoriali, etc.);

* Suggerimento A: sfruttare principi del genus-differentia;
* Suggerimento B: sfruttare tassonomia WordNet nell’esplorazione;
* Suggerimento C: pensare a meccanismi di backtracking.

**2.2 Svolgimento**

Inizialmente viene caricato il file *concept\_definitions.csv* contenente le definizioni relative ai termini. Ogni elemento della lista è l’insieme delle definizioni relative ad un termine. Per ogni set di definizioni di un termine viene applicato un pre-processing (tokenizzazione, rimozione stopwords e punteggiatura, lemmatizzazione) e vengono calcolati i termini più comuni attraverso la funzione getCommonTerms(). La funzione ritorna una lista ordinata in base alla frequenza dei termini presenti (calcolata con la funzione Counter(d).most\_common()).

Dopo aver calcolato i termini più frequenti nelle definizioni di un termine, vengono calcolati i synset associati ai 10 termini più frequenti (getSynsetsFromLemma()). Questo viene fatto perchè si ipotizza che i termini più frequenti sono quelli più rilevanti nella definizione del concetto.

La ricerca dei synset si basa sul meccanismo Genus-differentia, secondo il quale la descrizione di un concetto è composta da due parti:

* Genus, include il concetto da definire in una tassonomia, ovvero prevede di descrivere il concetto attraverso un suo iperonimo;
* Differentia, porzione della definizione che differenzia il concetto dal genus (discriminante).

In base a questo principio, la ricerca dei synset associati ai lemmi viene fatta anche sugli iperonimi e sugli iponimi. Inoltre viene fatta la stessa ricerca anche sui synset che sono iponimi e iperonimi, in maniera ricorsiva. La profondità di ricerca viene impostata con le variabili hyponyms\_limit e hypernyms\_limit. La funzione ritorna una lista di synset associati ai lemmi calcolati inizialmente.

Per determinare il synset più adatto alle definizioni si utilizza un approccio bag-of-words, nel quale si determinano due contesti e la similarità tra i due concetti viene calcolata come

Ovvero la cardinalità dell’insieme intersezione tra i due contesti più 1.

I due contesti sono:

* Lista di token ottenuta dal pre-processamento delle definizioni del termine (i lemmi);
* Lista di token ottenuta dal pre-processamento della definizione e degli esempi del synset (calcolati come s.definition() e s.examples()).

Il synset migliore è quello che ottiene un overlap maggiore con i lemmi delle definizioni.

I risultati sono elencati nella tabella che segue. Come si può notare, solo in un caso l’algoritmo mappa in maniera corretta il synset alle definizioni ('greed.n.01') mentre in 4 casi viene individuato il contesto corretto ('right.n.01', 'governed.n.01', 'carbohydrate.n.01', 'band.n.11'). Nel caso dei termini “patience”, “radiator” e “vehicle” il synset trovato non è corretto.

Nella seconda tabella è presente l’elenco ordinato dei 10 migliori synset per ogni termine. Come si può notare, molto spesso ci sono diversi synset con lo stesso valore massimo. Inoltre la variazione del punteggio è molto basso tra i primi dieci synset. Nel caso di termini come “radiator” e “vehicle”, il synset associato è rispettivamente Synset('hot.a.01') e Synset('container.n.01')( con punteggio 7 e 8), ma tra i primi dieci risultati ci sono anche synset più adatti come Synset('heating\_system.n.01') (5) e Synset('wheeled\_vehicle.n.01') (7).

Mentre nella maggior parte delle definizioni è stato analizzato il synset perfettamente corrispondente al termine, nel caso dei termini “patience” e “screw” i synset associati direttamente a questi termini non sono stati analizzati.

|  |  |  |
| --- | --- | --- |
| **Correct term** | **Synset** | **Synset definition** |
| justice | Synset('right.n.01') | an abstract idea of that which is due to a person or governmental body by law or tradition or nature; ; - Eleanor Roosevelt |
| patience | Synset('day.n.07' | the period of time taken by a particular planet (e.g. Mars) to make a complete rotation on its axis |
| greed | Synset('greed.n.01') | excessive desire to acquire or possess more (especially more material wealth) than one needs or deserves |
| politics | Synset('governed.n.01') | the body of people who are citizens of a particular government; --Declaration of Independence |
| food | Synset('carbohydrate.n.01') | an essential structural component of living cells and source of energy for animals; includes simple sugars with small molecules as well as macromolecular substances; are classified according to the number of monosaccharide groups they contain |
| radiator | Synset('hot.a.01') | used of physical heat; having a high or higher than desirable temperature or giving off heat or feeling or causing a sensation of heat or burning |
| vehicle | Synset('container.n.01') | any object that can be used to hold things (especially a large metal boxlike object of standardized dimensions that can be loaded from one form of transport to another) |
| screw | Synset('band.n.11') | a thin flat strip or loop of flexible material that goes around or over something else, typically to hold it together or as a decoration |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **justice** | | **patience** | | **greed** | |
| *Synset* | *Similarity* | *Synset* | *Similarity* | *Synset* | *Similarity* |
| 'right.n.01' | 6 | 'day.n.07' | 5 | 'greed.n.01' | 7 |
| 'human\_right.n.01' | 4 | 'digest.v.03' | 5 | 'acquisitiveness.n.01' | 5 |
| 'jurisprudence.n.01' | 4 | 'long\_run.n.01' | 4 | 'air.n.03' | 5 |
| 'rule.n.01' | 4 | 'stretch.n.06' | 4 | 'tone.n.10' | 5 |
| 'law.n.02' | 4 | 'lunar\_day.n.01' | 4 | 'recommendation.n.03' | 5 |
| 'entitlement.n.01' | 4 | 'able.s.03' | 4 | 'covet.v.01' | 5 |
| 'principle.n.04' | 4 | 'blue\_moon.n.01' | 4 | 'color.n.08' | 5 |
| 'use.n.07' | 4 | 'spontaneity.n.01' | 4 | 'hunger.n.02' | 5 |
| 'military\_law.n.01' | 4 | 'compassion.n.02' | 4 | 'magnificence.n.02' | 5 |
| 'abstraction.n.01' | 4 | 'mental\_quickness.n.01' | 4 | 'possessiveness.n.01' | 4 |
| 'justice.n.01' | 3 | Not evaluated |  | 'greed.n.01' | 7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **politics** | | **food** | | **radiator** | |
| *Synset* | *Similarity* | *Synset* | *Similarity* | *Synset* | *Similarity* |
| 'governed.n.01' | 6 | 'carbohydrate.n.01' | 5 | 'hot.a.01' | 7 |
| 'section.n.03' | 6 | 'biology.n.02' | 5 | 'central\_heating.n.01' | 6 |
| 'regulate.v.02' | 5 | 'animation.n.01' | 5 | 'utility.n.06' | 6 |
| 'relationship.n.03' | 5 | 'reservoir.n.04' | 5 | 'furnace\_room.n.01' | 6 |
| 'politics.n.05' | 5 | 'parasite.n.01' | 5 | 'heating\_system.n.01' | 5 |
| 'government.n.01' | 5 | 'life.n.11' | 5 | 'section.n.04' | 5 |
| 'authoritarian\_state.n.01' | 5 | 'parent.n.02' | 5 | 'mineral\_water.n.01' | 5 |
| 'population.n.04' | 5 | 'embryo.n.02' | 5 | 'position.n.07' | 4 |
| 'state.n.04' | 5 | 'process.n.05' | 5 | 'dining-hall.n.01' | 4 |
| 'utopia.n.02' | 5 | 'life.n.03' | 4 | 'component.n.03' | 4 |
| 'politics.n.02'  'politics.n.05' | 4  5 | 'food.n.01' | 4 | 'radiator.n.01' | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **vehicle** | | **screw** | |
| *Synset* | *Similarity* | *Synset* | *Similarity* |
| 'container.n.01' | 8 | 'band.n.11' | 7 |
| 'wheeled\_vehicle.n.01' | 7 | 'slice.n.05' | 6 |
| 'way.n.06' | 7 | 'solder.n.01' | 6 |
| 'component.n.03' | 6 | 'counter.n.08' | 6 |
| 'translocate.v.02' | 6 | 'beam.n.02' | 6 |
| 'motion.n.06' | 6 | 'section.n.04' | 6 |
| 'airlift.n.01' | 6 | 'connect.v.01' | 6 |
| 'handcart.n.01' | 6 | 'bimetal.n.01' | 5 |
| 'section.n.04' | 6 | 'join.v.02' | 5 |
| 'teleportation.n.01' | 5 | 'ridge.n.06' | 5 |
| 'vehicle.n.01' | 4 | Not evaluated |  |

**3. Esercitazione: Semantic clustering (P. Hanks)**

**3.1 Consegna**

L’esercitazione prevede l’implementazione della teoria di P. Hanks:

1. Scegliere un verbo transitivo (minimo valenza = 2);
2. Recuperare da un corpus n istanze in cui esso viene usato;
3. Effettuare parsing e disambiguazione
4. Usare i super sensi di WordNet sugli argomenti (subj e obj) del verbo scelto;
5. Aggregare i risultati, calcolare le frequenze, stampare i cluster semantici ottenuti.
   1. Un cluster semantico è inteso come combinazione dei semantic types (ad esempio coppie di sem\_types se valenza = 2)

**3.2 Svolgimento**

La teoria di Hanks prevede che il verbo sia la radice del significato. Una volta definita la valenza del verbo, ovvero il numero di argomenti che il verbo richiede, ad ogni argomento viene associato uno slot. Ogni slot può avere diversi valori, detti filler. Inoltre ad ogni *filler* è possibile associare dei *tipi semantici*, ovvero delle generalizzazioni concettuali. Il significato di un verbo dipende dai filler e dai tipi semantici ad esso associati.

Inizialmente vengono recuperate delle frasi dal Brown Corpus contenenti il verbo scelto (extractBrownSentences()). Per ogni frase estratta viene effettuato il parsing e si determinano gli argomenti associati al verbo. La funzione dependencyParsing() permette di calcolare l’albero a dipendenze tramite la risorsa spaCy [1]. Successivamente la funzione extractVerbSubjObj() determina il token relativo al verbo scelto e i suoi argomenti.

def extractVerbSubjObj (verb, tree):

subj\_dept = ['nsubj', 'nsubjpass']

obj\_dept = ['dobj', 'obj']

lemmatizer = WordNetLemmatizer()

verbAddress = next(t.text for t in tree if lemmatizer.lemmatize(t.text, 'v') == verb)

subjects= list(t.text for t in tree if str(t.head) == verbAddress and t.dep\_ in subj\_dept)

objects = list(t.text for t in tree if str(t.head) == verbAddress and t.dep\_ in obj\_dept)

return subjects, objects

In particolare:

* il verbo è il token dell’albero il cui lemma corrisponde al verbo scelto;
* il soggetto è il token il cui reggente (t.head) corrisponde a verbAddress e la cui relazione sintattica (t.dep\_) è ‘*nsubj*’ o ‘*nsubjpass*’;
* l’oggetto è il token il cui reggente (t.head) corrisponde a verbAddress e la cui relazione sintattica (t.dep\_) è ‘*dobj*’ o ‘*obj*’.

In questo modo è possibile determinare i filler (soggetti nominali e oggetti) del verbo scelto [2].

Nel caso in cui il verbo abbia valenza due, se sono presenti sia il soggetto che l’oggetto, avviene la disambiguazione dei filler attraverso l'algoritmo di Lesk. L'algoritmo di Lesk prende in input un termine polisemico e la frase in cui occorre e restituisce il senso migliore. In questo caso permette di determinare il miglior WordNet synset associato ad ogni filler, a partire dal filler e dalla frase in cui compare.

Si utilizza sia la funzione implementata in NLTK, sia una versione dell’algoritmo personale. Quest’ultima, per ogni senso associato al termine da disambiguare (ottenuto tramite wn.synsets(word)), calcola l'overlap tra i contesti della frase e del synset. I due contesti sono ottenuti con un approccio bag-of-words e sono composti da:

* ctx\_sentence: composto da tutti i termini della frase soggetti a pre-processing (tokenizzazione, rimozione punteggiatura e stopwords, lemmatizzazione).
* ctx\_synset: composto da tutti i termini presenti nella definizione e negli esempi soggetti a pre-processing (tokenizzazione, rimozione punteggiatura e stopwords, lemmatizzazione).

L'algoritmo ritorna il senso migliore, ovvero il synset che ha ottenuto l'overlap maggiore.

Nel caso in cui l’algoritmo di Lesk abbia determinato un senso, si determina il tipo semantico attraverso il suo supersenso (synset.lexname()) [3]. Nel caso in cui il soggetto o l’oggetto siano dei pronomi personali, viene associato il supersenso 'noun.person' o 'noun.object' (se il pronome è *“it”*).

Infine vengono calcolate le frequenze dei cluster semantici, ovvero la combinazione dei semantic types.

Sono stati fatti i test con i verbi *watch*, *play* e *get*, ottenendo i seguenti risultati

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Verb** | **Total sentences** | **Analyzed sentences**  **(valency = 2)** | **Semantic clusters** | **Most frequent cluster** |
| Watch | 197 | 53 | 27 | (‘noun.group’, ‘noun.person’) |
| Play | 308 | 117 | 43 | (‘noun.person’,’noun.act’) |
| Get | 1407 | 422 | 85 | (‘noun.person’, ‘noun.communication’) |

Di seguito l’istogramma relativo ai 15 semantic clusters più frequenti.

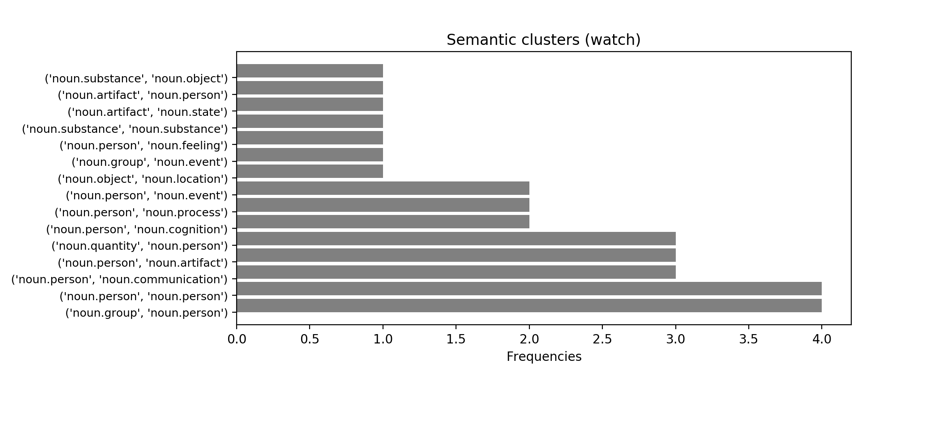


Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

**4. Esercitazione: Text segmentation**

**4.1 Consegna**

L’esercitazione prevede l’implementazione, ispirandosi al Text Tiling, di un algoritmo di segmentazione del testo:

1. Usando informazioni come frequenze (globali, locali), co-occorrenze, risorse semantiche (WordNet, etc.), applicando step di preprocessing (as usual), etc.

* La scelta del testo è a discrezione dello studente.

**4.2 Svolgimento**

**5. Esercitazione: OIE system**

**5.1 Consegna**

L’esercitazione prevede l’implementazione di un sistema di OIE (lezione 5 Giugno)

**5.2 Svolgimento**

**SITOGRAFIA**

[1] <https://spacy.io/usage/linguistic-features#dependency-parse>

[2] https://spacy.io/api/annotation

[3] https://wordnet.princeton.edu/documentation/lexnames5wn

[4] <https://github.com/ptorrestr/py_babelnet>

[5] <https://babelnet.org/guide>

<https://spacy.io/usage/linguistic-features>

<https://stackoverflow.com/questions/36610179/how-to-get-the-dependency-tree-with-spacy>

<https://spacy.io/usage/visualizers>

<https://spacy.io/api/token#attributes>

<https://spacy.io/api/annotation>

<https://stackoverflow.com/questions/39323325/can-i-find-subject-from-spacy-dependency-tree-using-nltk-in-python>