Tecnologie del Linguaggio Naturale

Esercitazioni del programma del Prof. Di Caro

Amedeo Racanati 928995

Angelo Pio Sansonetti 928869

Definitions lexical overlapping

Lo scopo di questa esercitazione è di valutare il grado di somiglianza delle definizioni che vengono date per esprimere un particolare concetto.

Per ogni definizione data per un concetto, effettuiamo un preprocessing rimuovendo le stop words e lemmatizzando le parole significative. Successivamente calcoliamo la similarità di ciascuna coppia di definizioni (rappresentate come due insiemi di parole), definita come il rapporto tra l'intersezione dei due insiemi e la cardinalità dell'insieme più piccolo.

In seguito si va a calcolare la media delle similarità calcolate, per ciascun concetto.

Di seguito è mostrato l'output ottenuto:

Similarity for definitions of concept building: 27%

Similarity for definitions of concept molecule: 18%

Similarity for definitions of concept freedom: 8%

Similarity for definitions of concept compassion: 10%

Mean similarity for 'concreto' concepts: 22%

Mean similarity for 'astratto' concepts: 9%

Mean similarity for 'generico' concepts: 17%

Mean similarity for 'specifico' concepts: 14%

Osservando i risultati, vediamo come siano più simili le definizioni per i concetti concreti piuttosto che per quelli astratti, in quanto per poter definire quest'ultimi aumenta la variabilità del lessico e quindi aumenta la difficoltà nel definirli. Infine abbiamo notato che i risultati ottenuti sono in linea con quelli mostrati da altri studenti a lezione.

Content-to-form

L'obiettivo di questa esercitazione è di inferire una serie di concetti a partire dalle loro definizioni. Quello che viene restituito è il synset in Wordnet.

Date le definizioni di un concetto, per trovare il relativo synset si effettuano i seguenti passi:

1. Partendo dalle definizioni, si effettua un preprocessing delle frasi e viene creato un array contenente i termini con la relativa frequenza;
2. Da questo array vengono presi i tre termini più frequenti e si calcolano i synset corrispondenti mediante l'algoritmo di Lesk, utilizzando come contesto di disambiguazione le stesse definizioni;
3. Per ogni synset si individuano i suoi iponimi, e per ognuno di essi si ottiene il contesto del synset (definito come l'insieme delle parole nella glossa e negli esempi) e si calcola la similarità di questo contesto contesto con l'insieme delle parole delle definizioni del concetto. La similarità è calcolata come il rapporto tra l'intersezione dei due insiemi e la cardinalità dell'insieme più piccolo;
4. Si restituisce il synset che massimizza la similarità descritta al passo precedente.

Prima di giungere a questa soluzione abbiamo provato ad effettuare una ricerca nel sottoalbero dell'iperonimo comune dei due termini più frequenti, ma il synset target individuato è stato quasi sempre lo stesso per ciascuno dei vari concetti.

Di seguito è mostrato l'output ottenuto:

Concetto da individuare: justice -> Best Synset Synset('legal_right.n.01')

Concetto da individuare: patience -> Best Synset Synset('endowment.n.01')

Concetto da individuare: greed -> Best Synset Synset('acquisitiveness.n.01')

Concetto da individuare: politics -> Best Synset Synset('operation.n.01')

Concetto da individuare: food -> Best Synset Synset('livestock.n.01')

Concetto da individuare: radiator -> Best Synset Synset('heater.n.01')

Concetto da individuare: vehicle -> Best Synset Synset('express.n.03')

Concetto da individuare: screw -> Best Synset Synset('solder.n.01')

Implementazione teoria di P. Hanks

L'obiettivo di questa esercitazione è di individuare per un particolare verbo i suoi cluster semantici, ovvero tutte le possibili combinazioni dei semantic types degli argomenti del verbo.

Il verbo scelto per l'esperimento è "say". Grazie al corpus Brown, è stato possibile ottenere per diecimila frasi il primo sostantivo a sinistra del verbo ed il primo sostantivo alla sua destra.

Una volta ottenute tutte le coppie di sostantivi, è stato individuato il relativo supersenso in Wordnet, aggregando e calcolando successivamente i conteggi per ciascuna coppia di supersensi.

La distanza tra i supersensi e la radice equivale a due.

Di seguito è mostrato l'output ottenuto:

39 per la coppia -> causal_agent.n.01 + psychological_feature.n.01

30 per la coppia -> group.n.01 + psychological_feature.n.01

27 per la coppia -> psychological_feature.n.01 + causal_agent.n.01

26 per la coppia -> causal_agent.n.01 + relation.n.01

20 per la coppia -> causal_agent.n.01 + causal_agent.n.01

19 per la coppia -> causal_agent.n.01 + communication.n.02

17 per la coppia -> causal_agent.n.01 + attribute.n.02

15 per la coppia -> causal_agent.n.01 + object.n.01

14 per la coppia -> group.n.01 + relation.n.01

13 per la coppia -> causal_agent.n.01 + group.n.01

12 per la coppia -> object.n.01 + psychological_feature.n.01

…

Dai risultati ottenuti è possibile notare che a sinistra del verbo say è spesso presente il supersenso *causal_agent*, ed è plausibile come risultato dato che chi deve parlare è quasi sempre un determinato agente (persona). A destra del verbo ci sono diversi supersensi, tra i quali spicca quello relativo ad una proprietà psicologica (*psychological_feature*).

Implementazione algoritmo di segmentazione ispirandosi al Text Tiling

L'obiettivo di questa esercitazione è di implementare un algoritmo di segmentazione del testo, utilizzando informazioni come frequenze (globali, locali), co-occorrenze, risorse semantiche (WordNet, etc.) e applicando step di preprocessing.

L'articolo dal quale ci siamo ispirati per sviluppare questo task è disponibile al seguente link: https://www.euronews.com/2020/06/11/brexit-draft-deal-first-of-many-hurdles-to-a-smooth-exit

L'algoritmo implementato prevede la suddivisione del testo in "finestre" di grandezza fissa, per ognuna delle quali viene effettuata la tokenizzazione delle parole. Successivamente viene calcolata per ciascuna coppia di finestre contigue lo score di dissimilarità, dato dalla sommatoria delle dissimilarità di ciascuna coppia di parole per le due finestre. La dissimilarità tra due parole è definita come *1 – similarità di WU & Palmer*.

Una volta calcolate tutte le cosiddette "window gaps", viene calcolata la media di questi valori e la deviazione standard. Sono ritenuti significativi quei gap che hanno uno score maggiore alla *media degli score + deviazione standard degli score*; abbiamo anche utilizzando una variante dove si considerano significativi quei gap che sono maggiori della *media + deviazione standard / 2*.

Abbiamo notato che l'algoritmo riesce a trovare nella maggior parte dei casi dei gap in corrispondenza delle "window" appartenenti a paragrafi differenti; tuttavia vi sono alcuni paragrafi che non si è riuscito a suddividere.

Implementazione della beam search per la generazione di nomi di band metal

L'algoritmo implementato per la generazione dei nomi di band metal in una RNN è quello relativo alla Beam Search, la quale memorizza solo le tre ipotesi migliori durante l'espansione dei possibili cammini. Ad ogni passo vengono espansi gli M cammini prendendo le N (N = M*2) parole (o caratteri) aventi più alta probabilità. Una peculiarità del nostro algoritmo è che qualora vengano trovate delle parole finite (ossia terminano con </s>), si prosegue ugualmente con l'espansione degli altri cammini non ancora conclusi, in quanto questi potrebbero avere una probabilità maggiore rispetto alle parole più corte trovate.

Una volta trovate tutte le parole finite, si restituiscono le migliori tre. Di seguito è mostrato l'output ottenuto.

Names generated after iteration 1000:

With beam search:

Astertion

Antertion

Astertiontion

With standard method:

cɜmorcClel

Aroriesdia

Acaron EnrAmty

Names generated after iteration 6000:

With beam search:

Death of Death

Death of Deathor

Death of Deather

With standard method:

Conocicy Velathrra

Dysrmal

Disonphat

Names generated after iteration 14000:

With beam search:

Morteration

Morteration Death

Morterations

With standard method:

Riceepent Monze

Morpa Infoulntio

Magthubhepser

Names generated after iteration 17000:

With beam search:

Recrotion

Recrotion Death

Recrotiony

With standard method:

Rieah

Sacroftation

Reth