Esercitazione Radicioni

Il primo problema che possiamo affrontare con lo strumento WordNet è la **“conceptual similarity”,** la somiglianza concettuale fra termini distinti. Il problema della conceptual similarity può essere inquadrato in questo modo:

- dati in input due termini, il task di conceptual similarity consiste nel fornire un punteggio numerico di similarità̀ che ne indichi la vicinanza semantica
- nel caso siano polisemici, (consideriamo che un minimo livello di polisemia tipicamente è sempre presente) bisogna anche definire in quali accezioni li stiamo confrontando

Ad esempio, la similarità̀ fra i concetti **car e bus** potrebbe essere **0.8** in una scala **[0,1]**, in cui:

- 0 significa che i sensi sono completamente dissimili
- 1 significa che i termini sono sinonimi.

Il problema della vicinanza semantica si studia perchè ha tante applicazioni, ad esempio per confrontare documenti, per capire se i documenti parlano di qualcosa che mi interessa, anche a dispetto del fatto che possono non occorrere gli stessi termini.
Ci sono diversi approcci a questo tak:

- Contare i termini che hanno in comune 2 sensi.
- Oppure guardiamo quanto sono sostituibili, ad esempio la distanza di hamming, tuttavia questa operazione opera a livello sintattico ma non semantico.
- Oppure proviamo ad usare WordNet:

Intuitivamente per risolvere il problema si può prendere la distanza tra i nodi dell’albero di wordNet associati ai 2 sensi, se hanno un genitore in comune. La somiglianza fra 2 concetti è in qualche modo una funzione della lunghezza del cammino che permette di collegare i 2 sensi in questione. Il problema è che in input abbiamo 2 termini non 2 synset. Ci può essere **ambiguità,** data dal fatto che un termine può avere più sensi.
Bisogna **considerare il contesto per compiere un’operazione di disambiguazione**, ad esempio la distanza semantica tra “calcio” e “ceffone” è alta, considerando il contesto della lotta, e parimenti lo è tra “calcio” e “magnesio” se ci mettiamo nel contesto degli elementi chimici.

Usiamo la risorsa WordSim353
WordSim353: è una risorsa largamente usata per risolvere questo task. Composta da 353 coppie di termini annotate da studenti laureati, utilizzati come testset in varie competizioni internazionali. A ciascuna coppia è attribuito un valore numerico [0,10], che rappresenta la similarità fra gli elementi della coppia.

PARTE1

IMPLEMENTARE LE METRICHE NON COPIARE
Lo scopo è implementare 3 metriche di similarità che sono basate su algoritmi in cui si cerca un path fra sensi, (basate su WordNet). Per ciascuna di tali misure di similarità, calcolare:

- gli indici di correlazione di Spearman
- gli indici di correlazione di Pearson

fra i risultati ottenuti e quelli 'target' presenti nel file annotato. Costituiscono entrambi dei rapporti di covarianza. Si usa Pearson quando si assume che i valori sono equamente distribuiti.

Metrica1: WU e PALMER:
La similarità è calcolata come 2* la profondità del più basso antenato comune fratto la somma della profondità del primo e del secondo.

Metrica2: Shortest Path
Prova a cercare un percorso minimo tra i 2 elementi. Calcolato come 2* profondità massima, (è una costante nota) - lunghezza del percorso dall'uno all'altro termine.

Metrica3: Leackon e chodorov
log (lunghezza del percorso tra t1 e t2) fratto 2 * profondità massima.

Quanti valori dobbiamo restituire? In input abbiamo
w1 e w2 e vogliamo un solo punteggio in output. Calcoliamo la massima similarity tra tutti i possibili accoppiamenti di sensi di w1 e w2.

PARTE2
DISAMBIGUAZIONE

WordSense Disambiguation

E' un problema IA del tipo: dato un termine polisemico e un contesto in cui il termine appare, si tratta di individuare e selezionare il senso inteso per quel termine in quel contesto. Si potrebbe generalizzare a tutti i termini della frase.

Capire quale synset di WordNet è associato al termine polisemico dato un contesto è un problema che sui dataset correnti si risolve con un'accuratezza che varia tra il 70 e l'80%.

Ci permette di capire che cosa parla un testo.

Esempio:

WordNet Sense	**Contesto**	**Frase**
BASS	**Musicale**	**play bass because it…**

Si può prendere tutto il contesto che si vuole ma da un certo punto in poi tende ad essere controproducente.

Approccio basato su vettori di features

Ci sono 2 modi per raccogliere features:

- **collocational**
- **bag of words**

collocational: Funziona prendendo a partire dal termine da disambiguare le parole affianco a distanza -n e +n, definito n qualsiasi, in modo simmetrico, usando come features il POS tag [Wi-2,POSi-2, Wi-1,POSi-1, Wi+1,POSi+1,Wi+2,POSi+2].

Esempio:

- Data la frase: "An electric guitar and bass player stand off to one side…." Vogliamo disambiguare il termine "bass".
- Usiamo l'approccio collocational scegliendo n = 2
- Otteniamo [guitar, NN, and, CC, player, NN, stand, VB]
- Almeno ¾ delle feature sono utili per disambiguare

bag of word: Approccio in cui destrutturiamo completamente la frase. La frase non ha più la sua sequenzialità e tutte le parole finiscono in un bag.

Il vettore che possiamo costruire è fatto da tutte le parole di contenuto (nomi, verbi, aggettivi, avverbi) più frequenti a partire da una collezione di frasi con il termine "bass" e contiene tanti 0 in corrispondenza delle parole che nel caso specifico non sono presenti nella frase e 1 invece per quelle presenti.

Esempio:

- 12 parole di contenuto più frequenti: [fishing, big, sound, player, fly, road, pound, double, runs, playing, guitar, band]
- vettore: [0,0,0,1,0,0,0,0,0,0,1,0] binario. possiamo avere anche i conteggi

Da una parte prendiamo informazioni sequenziali, dall'altra NO
L'algoritmo che prendiamo per fare WordSense Disambiguation è "Algoritmo di LESK"

the Lesk Algorithm

```
1  function SimplifiedLesk(word,sentence)
2  returns best sense of word
3  best-sense ← most frequent sense for word
4  max-overlap ← 0
5  context ← set of words in sentence
6  for all senses of word do
7     signature ← set of words in the gloss and examples of sense
8     overlap ← ComputeOverlap(signature,context)
9     if overlap > max-overlap then
10       max-overlap ← overlap
11       best-sense ← sense
12    end if
13 end for
14 return best-sense
```

Come funziona?

- Prende in input una coppia: il termine polisemico e la frase in cui occorre che funge da contesto.
- Viene inizializzato il senso migliore con il most frequent sense, ovvero il primo senso che viene restituito quando facciamo una ricerca su wordNet.
- Overlap massimo = 0
- Il contesto viene inizializzato con le parole nella frase in input
- A questo punto accediamo a WordNet. Prendiamo tutti i possibili sensi della parola in input.
- Per ogni senso creiamo una signature ovvero l'insieme dei termini nella glossa e negli esempi per quel senso li.
- Calcoliamo l'overlap tra il contesto polisemico e il contesto polisemico della signature
- Il resto del ciclo è dedicato a massimizzare l'overlap. Se l'overlap calcolato è maggiore di Max-overlap (inizialmente è a 0) allora assegno a questo l'overlap, best-sense = senso corrente.
- Ripeto per ogni senso
- restituisco il senso

Sostanzialmente restituisce il senso più frequente. Raggiunge il 60% di accuratezza.

IMPLEMENTAZIONE DELL'ALGORITMO DI LESK

1) Estrarre 50 frasi dal corpus SemCor
2) Randomizzare la selezione delle 50 frasi
3) Calcolare l'accuratezza media