Università degli Studi di Torino
Corso di Laurea Magistrale in Informatica
Tecnologie del Linguaggio Naturale
Docente: Daniele Radicioni

Relazione esercitazioni

Alessandro Clocchiatti
Matricola 909105

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

1. Conceptual similarity with WN
2. Word sense disambiguation
3. Mapping di Frame in WN Synsets (FrameNet)
4. Nasari summarization
5. Sense identification

1. Esercitazione: Conceptual similarity

1.1 Consegna

Dati in input due termini, il task di conceptual similarity consiste nel fornire un punteggio numerico di similarità che ne indichi la vicinanza semantica. Per esempio, la similarità 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, mentre 1 significa identità.
Per risolvere il task di conceptual similarity è possibile sfruttare la struttura ad albero di WordNet.

L'input per questa esercitazione è costituito da coppie di termini contenute nel file *WordSim353* (disponibile nei formati *.tsv* e *.csv*). Il file contiene 353 coppie di termini 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.

L'esercitazione consiste nell'implementare tre misure di similarità basate su WordNet. Per ciascuna di tali misure di similarità, calcolare gli indici di correlazione di Spearman e gli indici di correlazione di Pearson fra i risultati ottenuti e quelli *target* presenti nel file annotato.

Le tre misure di similarità sono:

- **Wu & Palmer**
  Questa metrica si basa sulla struttura di WordNet e la similarità tra due synset si calcola come
  $$cs(s1, s2) = \frac{2 \cdot depth(LCS)}{depth(s1) + depth(s2)}\ dove$$
  LCS: primo antenato comune (Lowest Common Subsumer) fra i synset $s1$ e $s2$.
  $depth(x)$: funzione che misura la distanza fra la radice di WordNet e il synset x.

- **Shortest Path**
  $$sim_{path}(s1, s2) = 2 \cdot depthMax - len(s1, s2)\ dove$$
  $depthMax$: profondità massima di WordNet (valore fisso)
  $len(s1, s2)$: lunghezza del percorso più breve che collega i due synset $s1$ e $s2$.

  La similarità tra i due sensi $s1$ e $s2$ è funzione della lunghezza del percorso più breve che collega i due synset:
  - Se $len(s1, s2) = 0$ allora $sim_{path}(s1, s2)$ assume il valore massimo, ovvero $2 \cdot depthMax$;
  - Se $len(s1, s2) = 2 \cdot depthMax$ allora $sim_{path}(s1, s2) = 0$

  Quindi il valore di similarità è compreso nell'intervallo $[0, 2 \cdot depthMax\ ]$

- **Leakcock & Chodorow**

$$sim_{LC}(s1, s2) = -\log \frac{len(s1, s2)}{2 \cdot depthMax}$$

Quando $s1$ e $s2$ hanno lo stesso senso, $len(s1, s2) = 0$. Per evitare $\log(0)$ si aggiunge 1 sia al numeratore che al denominatore. Quindi il valore di similarità è compreso nell'intervallo $[0, \log(2 \cdot depthMax + 1)\ ]$

Attenzione: l'input è costituito da coppie di termini, mentre la formula utilizza sensi.
Per calcolare la similarità fra due termini immaginiamo di prendere la massima similarity fra tutti i sensi del primo termine e tutti i sensi del secondo termine.
Quindi i due termini funzionino come contesto di disambiguazione l'uno per l'altro.
La massima similarità tra due termini si calcola come

$$sim(w_1, w_2) = \max_{c_1 \in s(w_1), c_2 \in s(w_2)} [sim(c_1, c_2)]\ \ dove$$

c_1, c_2: concetti che appartengono ai synset
w_1, w_2: termini

I coefficienti di correlazione di Pearson [1] e di Spearman [2] si calcolano come

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X} \sigma_{rg_Y}}\ dove$$

$\rho_{X,Y}$: coefficiente di correlazione di Pearson
$cov(X, Y)$: covarianza
σ_X: deviazione standard di X
rg_X: rango di X

1.2 Svolgimento

Inizialmente, per ogni coppia di parole, si determina la lista di synset associati al termine tramite la funzione NLTK `wn.synsets(word)`. Le liste di synset associati ai termini in input vengono utilizzate per calcolare la similarità massima tra i due concetti. Per ogni possibile combinazione di synset riferita ai due termini viene calcolata la similarità e viene memorizzato lo score di similarità maggiore.
Il risultato della prima parte è una lista contenente i vari score di massima similarità tra due termini.

Lo svolgimento di questa esercitazione ha previsto la creazione delle funzioni necessarie al calcolo della similarità tramite le tre metriche Wu & Palmer, Shortest Path e Leakcock & Chodorow. In particolare, vengono implementate le seguenti funzioni:

- `getCommonSubsumer()`
- `getLowestCommonSubsumer()`
- `shortestPath()`
- `depthPath()`

La funzione `getCommonSubsumer()` permette di determinare la lista degli antenati comuni ai due synset in input. Inizialmente vengono calcolati tutti gli iperonimi dei due synset (tramite metodo NLTK `syn._all_hypernyms`). La lista degli antenati comuni si ottiene facendo l'intersezione tra le liste degli iperonimi dei due synset.
La funzione `getLowestCommonSubsumer()` determina il primo antenato comune (il più specifico) ai due synset. Inizialmente viene calcolata la lista di antenati comuni (tramite `getCommonSubsumer()`) ai due synset e, siccome l'LCS corrisponde all'antenato più specifico, la funzione ritorna il synset che ha distanza maggiore dal root.
La funzione `shortestPath()` calcola la lunghezza del percorso più breve che collega due synset. Inizialmente vengono calcolati la lista degli antenati comuni e l'LCS. Nel caso in cui non ci siano antenati comuni non è possibile determinare un percorso, altrimenti viene calcolata la lista di percorsi che vanno dal synset al root (si utilizza la funzione NLTK `synset.hypernym_paths()`). Per ogni percorso contenente LCS (l'antenato più specifico comune ai due synset) viene calcolata la distanza parziale tra il synset e l'antenato comune (LCS). Il percorso minimo (*shortest path*) tra due synset è il valore minimo della somma tra le due distanze parziali

$$distance(synset_1, LCS) + distance(synset_2, LCS)$$

per tutti i path che contengono LCS.

Il valore della profondità massima di WordNet, necessaria per il calcolo della similarità con Shortest Path e Leakcock & Chodorow, corrisponde a 20 e viene calcolata come

```
max(max(len(hyp_path) for hyp_path in ss.hypernym_paths()) for ss in wn.all_synsets())
```

Dopo aver calcolato la similarità tra i termini è necessario calcolare la correlazione tra le similarità calcolate e quelle target (presenti nel file in input *WordSim353.csv*). Siccome le similarità target e quelle relative alle metriche Shortest Path e Leakcock & Chodorow hanno un range di valori diverso vengono normalizzate.
Gli indici di correlazione vengono calcolate tramite il coefficiente di Pearson e quello di Spearman e il risultato è il seguente:

Correlazione tra valori di similarità calcolati e quelli target

Similarity metric	**Spearman index**	**Pearson index**
Wu & Palmer	0.288	0.337
Shortest Path	0.165	0.289
Leakcock & Chodorow	0.318	0.289

2. Esercitazione: Worde sense disambiguation

2.1 Consegna

Implementare l'algoritmo di Lesk (non utilizzare l'implementazione esistente, e.g. NLTK).

1. Disambiguare i termini polisemici all'interno delle frasi del file *sentences.txt*; oltre a restituire i synset ID del senso (appropriato per il contesto), il programma deve riscrivere ciascuna frase in input sostituendo il termine polisemico con l'elenco dei sinonimi eventualmente presenti nel synset.
2. Estrarre 50 frasi dal corpus SemCor (corpus annotato con i synset di WN) e disambiguare almeno un sostantivo per frase. Calcolare l'accuratezza del sistema implementato sulla base dei sensi annotati in SemCor (SemCor disponibile all'URL http://web.eecs.umich.edu/~mihalcea/downloads.html)

2.2 Svolgimento

La prima parte di questa esercitazione prevede di determinare il synset ID del miglior senso associato al termine e di determinare i sinonimi presenti nel synset.
La funzione `lesk()` prende in input il termine e la frase completa (`sentence`). Siccome l'algoritmo di Lesk prevede il calcolo della sovrapposizione dei contesti della frase e del senso, inizialmente viene calcolato il contesto riferito alla frase (`context`).
La funzione `bagOfWord()` calcola il contesto con un approccio bag of words, il cui risultato è una lista di termini. In questo caso dal contesto della frase viene rimossa la punteggiatura, i termini vengono lemmatizzati e vengono rimosse le stopwords.

Successivamente, per ogni senso (*synset ID*) associato al termine (sensi calcolati tramite `wn.synsets(word)`), viene calcolato il contesto del senso (`signature`). Il contesto di un synset, calcolato sempre con la funzione `bagOfWord()`, è una lista dei termini presenti nella definizione e negli esempi presenti nel synset (calcolati tramite `sense.definition()` e `sense.examples()`).
La sovrapposizione (*overlap*) tra i due contesti viene calcolata con la funzione `computeOverlap()`, la quale calcola la dimensione dell'insieme intersezione dei due contesti. Il miglior senso è quello che ottiene un overlap maggiore con il contesto della frase.
Per determinare i sinonimi di un synset ID si utilizza la funzione `findSynonym()`, la quale ritorna la lista di tutti i lemmi associati al synset (calcolati come `sense.lemma_names()`).

Il risultato, presente nel file *outputDisambiguation.csv*, è il seguente:

Original sentence	Word	Synset	Synonym list
arms bend at the elbow	arms	Synset('arm.n.01')	['arm']
germany sells arms to saudi arabia	arms	Synset('arm.n.02')	['arm', 'branch', 'limb']
the key broke in the lock	key	Synset('key.v.02')	['key']
the key problem was not one of quality but of quantity	key	Synset('key.v.05')	['key']
work out the solution in your head	solution	Synset('solution.n.01')	['solution']
heat the solution to 75° celsius	solution	Synset('solution.n.01')	['solution']
the house was burnt to ashes while the owner returned	ashes	Synset('ash.n.03')	['ash']
this table is made of ash wood	ash	Synset('ash.n.03')	['ash']
the lunch with her boss took longer than she expected	lunch	Synset('lunch.n.01')	['lunch', 'luncheon', 'tiffin', 'dejeuner']
she packed her lunch in her purse	lunch	Synset('lunch.n.01')	['lunch', 'luncheon', 'tiffin', 'dejeuner']
the classification of the genetic data took two years	classification	Synset('categorization.n.03')	['categorization', 'categorisation', 'classification', 'compartmentalization', 'compartmentalisation', 'assortment']
the journal science published the classification this month	classification	Synset('categorization.n.03')	['categorization', 'categorisation', 'classification', 'compartmentalization', 'compartmentalisation', 'assortment']
his cottage is near a small wood	wood	Synset('wood.n.08')	['wood']
the statue was made out of a block of wood	wood	Synset('wood.n.08')	['wood']

La seconda parte dell'esercitazione prevede la disambiguazione di 50 frasi (selezionate casualmente) dal file XML *br-a01.xml*. Per ogni frase, viene creata una lista (`tagged_words`) formata dai sostantivi presenti nella frase e viene selezionato in modo casuale un termine dalla lista (`word`).
Per determinare il Semcor synset associato al termine scelto è necessario utilizzare la funzione NLTK `wn.synset_from_sense_key()` [3], la quale ritorna il synset data una `sense_key`. Le chiavi di senso si ottengono tramite il lemma e il suo senso (`word.get("lemma"), word.get("lexsn")`) e hanno la seguente forma

```
lemma % lex_sense
```

Esempio: `highway%1:06:00:: , state%1:14:01::`

Successivamente si determina il synset ID del migliore senso associato al termine grazie all'algoritmo di Lesk, in modo da confrontare l'annotazione Semcor con l'algoritmo. In questo caso si utilizza sia l'algoritmo illustrato precedentemente (quello creato personalmente) sia l'implementazione di NLTK.
I sensi ottenuti vengono memorizzati nel file *outputSemcor.csv*. Di seguito alcuni esempi formattati:

Sentence	The Highway_Department source told The_Constitution , however , that Vandiver has not been consulted yet about the plans to issue the new rural roads bonds .		
Ambiguous term	**Semcor synset**	**My synset**	**NLTK synset**
source	Synset('beginning.n.04')	Synset('beginning.n.04')	Synset('source.n.07')

Sentence	Nevertheless , `` we feel that in_the_future Fulton_County should receive some portion of these available funds " , the jurors said .		
Ambiguous term	**Semcor synset**	**My synset**	**NLTK synset**
funds	Synset('investment_company.n.01')	Synset('store.n.02')	Synset('store.n.02')

Infine viene effettuato il confronto tra i sensi Semcor e quelli ottenuti tramite algoritmo di Lesk (personale e NLTK). Il confronto viene effettuato con la funzione `accuracy_score` di `sklearn.metrics`. Siccome sia la scelta delle frasi che il termine da disambiguare è casuale, i punteggi di accuratezza variano ad ogni esecuzione. In generale è stato notato che il punteggio risulta molto simile rispetto a quello ottenuto confrontando i Semcor synset con i sensi ottenuti da Lesk NLTK. Inoltre l'accuratezza dell'annotazione effettuata rispetto a quella di Semcor risulta essere compresa tra il 30% e il 50%.

3. Esercitazione: Mapping di Frame in WN Synsets

3.1 Consegna

La terza esercitazione prevede:

1. Come prima operazione ciascuno deve individuare un insieme di frame (nel seguito riferito come FrameSet) su cui deve lavorare, attraverso la funzione `getFrameSetForStudent(cognome)`. La funzione restituisce, dato un cognome in input, l'elenco di frame da elaborare.
2. Per ogni frame nel FrameSet è necessario assegnare un WN synset ai seguenti elementi:
   - *Frame name* (nel caso si tratti di una multiword expression (es. "Religious_belief") disambiguare il termine principale, che in generale è il sostantivo se l'espressione è composta da NOUN+ADJ, e il verbo se l'espressione è composta da VERB+NOUN;
   - *Frame Elements* (FEs) del frame;
   - *Lexical Units* (LUs).

   I contesti di disambiguazione possono essere creati utilizzando le definizioni disponibili (sia quella del frame, sia quelle dei FEs), ottenendo `Ctx(f)`, il contesto per FN terms f. Per quanto riguarda il contesto dei sensi presenti in WN è possibile selezionare glosse ed esempi dei sensi, e dei loro rispettivi iponimi e iperonimi, in modo da avere più informazione, ottenendo quindi il contesto di disambiguazione `Ctx(s)`.

   Il mapping può essere effettuato utilizzando (almeno) uno fra i due approcci:
   - Approccio bag of words;
   - Approccio grafico.
3. L'ultima parte prevede la valutazione dell'output del sistema.
   La correttezza dell'output del sistema sviluppato è da calcolare in rapporto all'annotazione effettuata manualmente. Quindi l'annotazione costituisce un elemento molto importante nello svolgimento dell'esercitazione.
   Il programma implementato dovrà quindi fornire anche la funzionalità di valutazione, che confronterà i synset restituiti in output dal sistema con quelli annotati a mano dallo studente; su questa base deve essere calcolata l'accuratezza del sistema, semplicemente come rapporto dei corretti sul totale.

3.2 Svolgimento

Il FrameSet individuato in base al cognome (Clocchiatti) è il seguente

```
student: Clocchiatti
        ID:   31        frame: Scrutiny
        ID:  120        frame: Arraignment
        ID: 1030        frame: Remainder
        ID: 1771        frame: Thriving
        ID: 2303        frame: Container_focused_placing
```

Successivamente, per ogni frame presente nel FrameSet, si determinano i seguenti elementi:

- Frame name (`f_name`, funzione `fn.frame_by_id(id).name`);
- Frame elements (`f_FE`, funzione `fn.frame_by_id(id).FE`);
- Lexical unit (`f_LU`, funzione `fn.frame_by_id(id).lexUnit`).

Siccome il mapping tra Frame e WN Synset prevede solamente l'utilizzo dell'approccio bag of words, viene calcolato il contesto del frame (`getFrameContext()`), il quale è formato dalla lista di termini che compongono la definizione del frame e delle definizioni dei frame elements del frame. Il contesto del frame (`ctx_frame`) resta lo stesso mentre cambia il contesto dei vari synset.
La funzione `getWNSynset()` permette di determinare il synset migliore per il frame sulla base dei tre elementi (frame name, frame elements, lexical units). In particolare, la funzione ritorna:

- Il miglior synset determinato in base al frame name;
- Per ogni frame element, il miglior synset determinato a partire dal singolo frame element (si ottiene quindi un synset per ogni frame element del frame);
- Per ogni lexical unit, il miglior synset calcolato a partire dalla singola lexical unit (si ottiene quindi un synset per ogni lexical unit del frame).

Synset – Frame name

La funzione `syn_frameName()` prende in ingresso l'ID del frame, il suo contesto e il frame name.
Nel caso in cui il frame name sia una multiword expression (se contiene il carattere "_"), è necessario disambiguare il termine principale. Per fare questo si calcola il part-of-speech del frame name diviso in token e il termine principale corrisponde al verbo o al nome.
Per ogni synset associato al frame name (synsets ottenuti tramite funzione `wn.synsets(frame_name)`) viene calcolato il contesto del synset e l'overlap tra i due contesti (contesto del frame e contesto del synset). Quest'ultimo viene calcolato come

$$score(s, w) = |Ctx(s) \cap Ctx(w)| + 1$$

Il contesto del synset (calcolato tramite `getSynsetContext()`) è composto dai seguenti elementi:

- Termini della definizione processata del synset;
- Termini degli esempi processati del synset;
- Termini delle definizioni e degli esempi processati degli iperonimi;
- Termini delle definizioni e degli esempi processati degli iponimi.

Il processamento prevede: rimozione delle stop-word, rimozione della punteggiatura e lemmatizzazione.

Il miglior synset dato il frame name è quello che ottiene l'overlap maggiore tra i due contesti.

Synset – Frame elements

La funzione `syn_frameElements()` prende in ingresso l'ID del frame, il suo contesto e i frame elements.
Per ogni frame element, vengono determinati i synset associati. Per ogni synset si calcola il rispettivo contesto (calcolato come descritto precedentemente) e viene calcolato l'overlap tra i contesti. Il synset che ottiene l'overlap maggiore è quello che mappa meglio il frame rispetto quel frame element.
Questa funzione, a differenza di quella precedente, ritorna una lista in cui, per ogni frame element, viene associato il miglior synset calcolato.

Synset – Lexical units

La funzione `syn_lexicalUnits()` prende in ingresso l'ID del frame, il suo contesto e le lexical units.
Per ogni lexical unit del frame, vengono determinati i synset corrispondenti. Per ogni synset si calcola il rispettivo contesto e viene calcolato l'overlap con il contesto del frame. Il synset che massimizza l'overlap è il miglior senso data una precisa lexical unit.
Come nel caso dei frame elements, la funzione ritorna una lista in cui, per ogni lexical unit, viene associato il frame al miglior synset.

La seconda parte dell'esercitazione prevede la valutazione dell'annotazione manuale rispetto ai risultati ottenuti con l'approccio bag of words.
L'annotazione manuale prevede il mapping tra il frame e il synset più appropriato sulla base dell'elemento del frame (frame name, frame element e lexical unit). Il synset più appropriato è stato scelto tra tutti quelli che il programma valuta (tutti i synset ritornati dalla funzione `wn.synsets(el)`). Per determinare quale synset sia migliore sono state utilizzate le definizioni dei synset.
Il confronto tra l'annotazione manuale e quella automatica viene fatta grazie alla funzione `compareAnnotation(man_ann,aut_ann)`, la quale valuta se le associazioni nei due file sono uguali o no. La funzione ritorna lo `score`, calcolato come

$$associazioni\ corrette\ /\ numero\ totale\ di\ elementi$$

ottenendo un'accuratezza del 65,3%.

4. Esercitazione: Nasari summarisation

4.1 Consegna

L'esercitazione prevede di implementare un algoritmo estrattivo che permette di ridurre le dimensioni del documento del 10, 20 e 30% seguendo i seguenti step:

1. Individuazione dell'argomento (topic) del testo da riassumere. L'argomento può essere indicato come un insieme di vettori Nasari.

$$Vt_1 = \{term_1\_score, term_2\_score, \dots, term_{10}\_score\ \}$$
$$Vt_2 = \{term_1\_score, term_2\_score, \dots, term_{10}\_score\ \}$$

2. Creazione del contesto, raccogliendo i vettori dei termini (questo passaggio può essere ripetuto, scaricando il contributo dei termini associati ad ogni ciclo);
3. Conservare i paragrafi contenenti i termini più salienti in base alla **Weighted Overlap** `WO(v1, v2)` Determinare il peso dei paragrafi applicando almeno uno degli approcci citati (titolo, spunto, frase, coesione).

Vengono forniti due file Nasari:

- *dd-nasari.txt* , un sottoinsieme di NASARI (ottenuto troncando i vettori a 10 caratteristiche). 3.587.754 vettori, ~ 600 MB (https://goo.gl/85BubW);
- *dd-small-nasari-15.txt* , un sottoinsieme di NASARI. È stato applicato lo stesso filtro di *dd-nasari.txt*, con 15 caratteristiche più l'intersezione con i lemmi 60K nel Corpus of Contemporary American English: 13.084 vettori, 2MB di archiviazione (in questo file molte entità sono state rimosse).

Il secondo file è stato estratto per iniziare la nostra sperimentazione, mentre il secondo ha lo scopo di esplorare la risorsa in maniera più approfondita.

I documenti da riassumere sono:

- *Andy-Warhol.txt*
- *Ebola-virus-disease.txt*
- *Life-indoors.txt*
- *Napoleon-wiki.txt*

Effettuare delle sperimentazioni con diversi livelli di compressione ($10\%, 20\%\ e\ 30\%$).

4.2 Svolgimento

La sintesi del documento in input si basa sul contenuto del titolo (*title method*) e si ottiene con la funzione `summarization()`.

Per determinare i topic del titolo si utilizza la funzione `getNasariVectors()`, la quale inizialmente tokenizza e processa la frase in input (rimozione stopwords e punteggiatura, lemmatizzazione) e poi, per ogni termine della frase ripulita (tokenizzata), si effettua la ricerca di quel termine nel dizionario Nasari. Il topic di una frase (ottenuto tramite `getNasariVectors()`) è composto dalla lista di vettori Nasari.

Per ogni paragrafo dell'articolo viene calcolato l'insieme dei topic (tramite `getNasariVectors()`). Per ogni argomento del paragrafo e per ogni argomento del titolo viene calcolato la sovrapposizione pesata (**Weighted Overlap**, tramite `getWeightedOverlap(c,t)`). La rilevanza di un paragrafo è calcolata come la media dei punteggi di overlap per ogni topic.

La sovrapposizione pesata tra due vettori Nasari si calcola determinando inizialmente l'insieme di chiavi riferite al vettore Nasari (`vect.keys()`) in comune ai due vettori Nasari. Nel caso in cui ci siano delle chiavi comuni ai due vettori, la Weighted Overlap si calcola come

$$WO(v_1, v_2) = \frac{\sum_{q \in O}\left(rank(q, v_1) + rank(q, v_2)\right)^{-1}}{\sum_{i=0}^{|O|}(2i)^{-1}}\ dove$$

O: insieme Overlap delle chiavi comuni ai due vettori
q: chiave contenuta nell'insieme delle chiavi comuni ai due vettori
$rank(q, v_1)$: determina la posizione della chiave q nel vettore Nasari, in modo da determinare la sua rilevanza in quel vettore. Questo dipende dal fatto in quanto i vettori Nasari sono ordinati in maniera decrescente in base alla rilevanza

In base al tasso di compressione, viene calcolato il numero di paragrafi che è necessario rimuovere. I paragrafi che ottengono una rilevanza minore vengono rimossi e il testo riassunto salvato.

5. Esercitazione: Sense identification

5.1 Consegna

La quinta esercitazione prevede due parti: semantic word similarity e sense identification.

Semantic word similarity:

1. Individuazione di 50 coppie di termini dal file *it.test.data.txt* sulla base del cognome (utilizzare la funzione presente nel notebook *semeval_mapper.ipynb*).
2. La prima operazione consiste nell'annotare con punteggio di semantic similarity 50 coppie di termini. Il criterio da utilizzare è il seguente (https://tinyurl.com/y6f8h2kd):

   4: Very similar - The two words are synonyms (*e.g., midday-noon*).
   3: Similar - The two words share many of the important ideas of their meaning but include slightly different details.They refer to similar but not identical concepts (*e.g., lion- zebra*).
   2: Slightly similar - The two words do not have a ver y similar meaning, but share a common topic/domain/function and ideas or concepts that are related (*e.g., house-window*).
   1: Dissimilar - The two items describe clearly dissimilar concepts, but may share some small details, a far relationship or a domain in common and might be likely to be found together in a longer document on the same topic (*e.g., software-keyboard*).
   0: Totally dissimilar and unrelated - The two items do not mean the same thing and are not on the same topic (*e.g., pencil-frog*).

   L'output della prima consegna è un file (in formato *.tsv*) di 50 linee, ciascuna contenente un numero in $[0,4]$, con la seguente struttura

   ```
   [word1, word2, similarity]
   ```
3. La valutazione dei punteggi annotati dovrà essere condotta in rapporto alla similarità ottenuta utilizzando i vettori NASARI (versione embedded, file *mini_NASARI.tsv*, nel materiale della lezione). La similarità tra due termini si determina tramite massimizzazione della cosine similarity

   $$sim_{cos}(\overrightarrow{V_1}, \overrightarrow{V_2}) = \frac{\overrightarrow{V_1} \cdot \overrightarrow{V_2}}{\|\overrightarrow{V_1}\|\|\overrightarrow{V_2}\|} \; dove$$

   $\overrightarrow{V_1}$: rappresentazioni vettoriali corrispondenti al senso c_1
   $\overrightarrow{V_1} \cdot \overrightarrow{V_2}$: prodotto delle distanze Euclidee dei due vettori di lunghezza N
   $\|\overrightarrow{V_1}\|$: norma 2 del vettore

4. La valutazione della nostra annotazione è condotta calcolando i coefficienti di Pearsons e Spearman fra (la media dei) i punteggi annotati a mano e quelli calcolati con la versione embedded di NASARI.
   Questa valutazione è sostanzialmente diversa dalla verifica di agreement nell'annotazione:
   - nell'inter-rater agreement verifichiamo se gli annotatori umani hanno annotato correttamente i vari elementi, e se il task è chiaramente formulato/risolvibile univocamente da esseri umani;
   - La valutazione consiste invece nel calcolare il livello di correlazione fra giudizio umano e punteggi calcolati algoritmicamente (massimizzazione sim_{cos} dei vettori).

Sense identification:

1. Il secondo compito consiste nell'individuare i sensi selezionati nel giudizio di similarità.
   La domanda che ci poniamo è la seguente: *quali sensi abbiamo effettivamente utilizzato quando abbiamo assegnato un valore di similarità a una coppia di termini (per esempio, società e cultura)?*
2. L'output di questa parte dell'esercitazione consiste in 2 Babel synset ID e dai termini del synset. Il formato dell'output ha la seguente struttura

   ```
   #Term1 Term2 BS1 BS2 Terms_in_BS1 Terms_in_BS2

   macchina bicicletta bn:00007309n bn:00010248n
   auto,automobile,macchina bicicletta,bici,bike
   ```

3. Calcoliamo nuovamente il livello di agreement nelle annotazioni, questa volta utilizzando il punteggio kappa di Cohen (è possibile utilizzare il `cohen_kappa_score` della libreria Python `sklearn.metrics`)

5.2 Svolgimento

Semantic word similarity

Le coppie individuate in base al cognome (Clocchiatti) sono

```
Clocchiatti     :        coppie nell'intervallo 201-250
```

Di seguito un estratto dell'annotazione manuale:

```
terremoto     scossa        3.5
patrimonio    azione        1
ebreo         Gerusalemme   2.5
nuvolosità    previsione    3
dizionario    enciclopedia  3.5
zecca         museo         0.5
sedia         sgabello      3.5
spagnolo      umidità       0
lattina       bottiglia     3.5
```

Il calcolo della similarità tra due termini in base ai vettori Nasari viene fatto con la funzione `getNasariScore()`. Per ogni coppia di termini viene calcolata la lista di sensi Babel associati ad ogni parola (funzione `getBabelId()`). Se le liste di sensi dei due termini non sono vuote, si calcola la similarità massima tra i sensi dei due termini.
La funzione `bestSenseSimilarity()` associa inizialmente ad ogni senso Babel il corrispondente vettore Nasari. Per ogni tupla `(sense,nasari_vector)` del primo termine e per ogni tupla `(sense,nasari_vector)` del secondo termine, viene calcolata la cosine similarity tra i due vettori Nasari. La funzione ritorna gli ID dei sensi che hanno ottenuto la similarità massima con il corrispondente score.

L'output della funzione `getNasariScore()` è una lista (`nasari_score`) di liste contenenti i termini e il corrispondente punteggio di similarità, ovvero

```
[[word1, word2, score]]
```

Infine, vengono estratte e normalizzate le liste contenenti i punteggi di similarità (sia quelli annotati manualmente sia quelli calcolati tramite cosine similarity) e viene calcolato l'indice di correlazione tra i due punteggi in base ai coefficienti di Pearson e di Spearman, ottenendo il seguente risultato:

```
Pearson index correlation: 0.717
Spearman index correlation: 0.758
```

Sense identification

Siccome la seconda consegna prevede un output con il seguente formato

```
[word1, word2, id_sense_word1, id_sense_word2, Terms_in_BS1 Terms_in_BS2]
```

si utilizza la funzione `getNasariScoreSenses()`, la quale è identica alla funzione `getNasariScore()` ma permette di ottenere una lista con il seguente formato

```
[word1, word2, id_sense_word1, id_sense_word2]
```

Per determinare i termini associati al miglior senso delle due parole (`id_sense_word1`) si utilizza la funzione `getBabelTerms()`, la quale permette di estrarre i Babel senses associati ad un babel ID. Per fare questo si utilizzano le API BabelNet [4][5]. La funzione `extractBabelTermAPI()` ottiene i sensi associati all' ID Babel nel seguente modo

```
api = BabelnetAPI(babel_key)
senses = api.get_synset(id=babel_id, targetLang="IT")
```

e successivamente recupera il lemma di ogni senso ottenuto (`i['properties']['fullLemma']`, dove `i` è un dizionario che caratterizza il senso).
Siccome l'utilizzo delle API BabelNet è limitato (1000 richieste al giorno), è stato creato un file offline contenente il mapping tra la lista dei termini e l'ID Babel (file *babelInfo_API.txt*).
Dopo aver salvato la lista contentente i termini, i migliori sensi Babel e i termini associati (babelList.txt), viene calcolato l'agreement nell'annotazione utilizzando il punteggio Kappa di Cohen (si utilizza la funzione presente in `sklearn.metrics`), ottenendo il seguente risultato:

```
Kappa Cohen score: 0.237
```

SITOGRAFIA

[1] https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[2] https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient
[3] http://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html#WordNetCorpusReader.synset_from_sense_key
[4] https://github.com/ptorrestr/py_babelnet
[5] https://babelnet.org/guide