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.
    a. 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, ogni *filler* può rappresentare un *semantic type*, ovvero una generalizzazione concettuale. Il significato di un verbo dipende dai filler e dai tipi semantici ad esso associati.

Inizialmente vengono recuperate delle frasi dal SEMCOR corpus che contengono il verbo scelto (questa operazione viene fatta solo una volta). Per ogni frase estratta viene effettuato il parsing e si determinano le dipendenze attraverso la funzione `getTree()` , che utilizza la risorsa spaCy. Successivamente la funzione `extractVerbSubjObj()` determina i token relativi al soggetto ed all'oggetto del verbo in ogni frase estratta dal corpo:

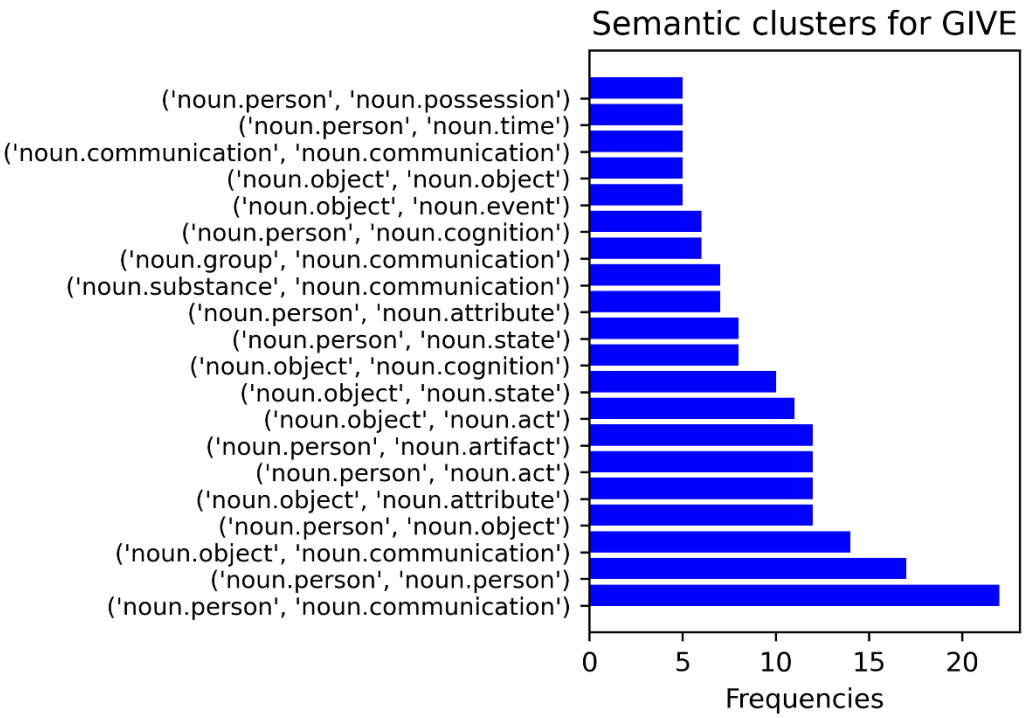
```
def extractVerbSubjObj (verb, sentences):
    subj_dept = ['nsubj', 'nsubjpass']
    obj_dept = ['dobj', 'obj']
    lemmatizer = WordNetLemmatizer()
    subj_obj_list = []
    ind = 0
    for s in sentences:
        tree = getTree(s)
        verbAddress = next(t.text for t in tree if lemmatizer.lemmatize(t.text, '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)
        subj_obj_list.append([subjects, objects, [ind]])
        ind += 1
    return subj_obj_list
```

In particolare:

- il verbo (`verbAddress`) 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.
Viene quindi creata una lista che contiene il soggetto, l'oggetto e l'indice della frase a cui appartengono. Viene effettuata una pulizia dei dati per eliminare tutte le frasi in cui il verbo non ha valenza 2, andando a prendere solo gli elementi della lista che hanno lunghezza uguale a 3.
Attraverso la funzione `getSemanticTypes()` viene associato ad ogni elemento soggetto ed elemento oggetto nei due slot di ogni frase il suo supersenso, ossia una delle 25 radici dei nomi presenti all'interno dell'albero di wordnet. Il metodo per prima cosa applica l'algoritmo di lesk per determinare il senso più corretto della parola per una determinata frase. In seguito attraverso la funzione `synset.lexname()` di wordnet viene restituito il supersenso e viene creata una lista di tuple: (supersenso del soggetto, supersenso dell'oggetto). Nel caso in cui l'algoritmo di lesk non trovi un senso viene applicato il supersenso '*noun.object',* che viene applicato anche se il soggetto sia un pronome indefinito (*it*), mentre, nel caso di pronomi personali alla tupla verrà aggiunto '*noun.person'*.
Infine, vengono calcolate le frequenze dei cluster semantici, ovvero la combinazione dei semantic types.

Di seguito l'istogramma relativo ai 20 semantic clusters più frequenti.



Nel file *results.txt* sono presenti altri dati come il numero totale di frasi con il verbo GIVE all'interno di SEMCOR, il numero totale di frasi effettivamente analizzate (quelle in cui il verbo ha valenza 2) ed il numero totale di semantic types trovati.