Esercitazione 2

Mapping di Frame in WN Synsets

Consegna

Per ogni frame nel FrameSet è necessario assegnare un WN synset ai seguenti elementi:

- **Frame name** (nel caso si tratti di una multiword expression, come per esempio '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; in generale l'elemento fondamentale è individuato come il **reggente dell'espressione**.)
- **Frame Elements (FEs)** del frame; e
- **Lexical Units (LUs)**.

Creazione Contesto Ctx(w)

"I contesti di disambiguazione possono essere creati utilizzando le definizioni disponibili (sia quella del frame, sia quelle dei FEs), ottenendo `Ctx(w)`, il contesto per FN terms `w`."

in code:

```
…
word1 = re.sub(r"[^a-zA-Z0-9 ]", "", local.definition).split()
word2 = []

for elem in local.FE:
   val = local.FE[elem]
   word2 = word2 + (re.sub(r"[^a-zA-Z0-9 ]", "",
val.definition).split())
word.append(set(word1 + word2))
…
```

Prendiamo le varie parole della definizione del frame e le inseriamo, ragionamento identico per i vari Frame Elements.

Nota: *invece di applicare un controllo ad ogni inserimento per i possibili valori duplicati, al termine di tutti gli inserimenti trasformiamo la nostra lista in un set in modo da eliminare possibili valori duplicati.*

Creazione contesto Ctx(s)

"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)`"

in code:

```
for sense in senses_list:
   elem_ctx = []
   for example in sense.examples():
       elem_ctx = elem_ctx + example.split()
   for glos in sense.definition().split():
       elem_ctx = elem_ctx + re.sub(r"[^a-zA-Z0-9]", "", glos).split()
   for hypo in sense.hyponyms():
```

```
        for example in hypo.examples():
            elem_ctx = elem_ctx + example.split()
        for glos in hypo.definition().split():
            elem_ctx = elem_ctx + re.sub(r"[^a-zA-Z0-9]", "", glos).split()
    for hyper in sense.hypernyms():
        for example in hyper.examples():
            elem_ctx = elem_ctx + example.split()
        for glos in hyper.definition().split():
            elem_ctx = elem_ctx + re.sub(r"[^a-zA-Z0-9]", "", glos).split()
```

Approcci al mapping

Abbiamo sviluppato i due possibili approcci al mapping dei sensi:

- **Approccio a bag of words**
- **Approccio Grafico**

Approccio Bag of Words

Scelta del senso che permette di massimizzare l'intersezione fra i contesti. In questo caso lo score è calcolato come:

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

dove verrà selezionato l'argomento S che massimizza lo score, ossia:

$$BestSense\ 's'\ per\ Word\ 'w' = argmax\ |Ctx(s) \cap Ctx(w)| + 1$$

in code:

```
overlap = len(set(elem_ctx).intersection(context))
if overlap >= max_overlap_FE:
    max_overlap_FE = overlap
    best_sense = sense
```

Approccio Grafico

In questo caso si procede con la costruzione di un grafo che contiene tutti i synset associati ai termini in Framenet (FN):

$$FN = w(FE) \cup w(LU)$$

```
while toDo:
    actual = toDo.pop()
    if not actual in done:
```

```
        len = actual.shortest_path_distance(start)
        if isinstance(len, int) and not len > 1:
            for child in actual.hyponyms():
                graph.add_edge(actual.name(), child.name())
                toDo.add(child)
            for parent in actual.hypernyms():
                graph.add_edge(actual.name(), parent.name())
                toDo.add(parent)
    done.add(actual)
return graph
```

Lo *score(s,w)* è calcolato sfruttando i synset associati ai termini da mappare e al loro contesto di disambiguazione. In particolare, si tratta di costruire il sottografo di *WN* contenente i synset presenti in tutti i path di lunghezza *l* ≤ *L* fra i possibili sensi dei termini del contesto di disambiguazione e *s*; la funzione che valuta l'importanza del senso *s* per il termine *w* calcola la seguente funzione:

$$score(s,w) = \sum_{cw \in Ctx(w)} \sum_{s' \in senses\ WN(cw)} \sum_{p \in path(s,s')} e^{-(len(p)-1)}$$

in code:

```
sum = 0
for term in ctx:
   for sense in wn.synsets(term):
       if graph.has_node(sense.name())
       and graph.has_node(poss_sense.name())
       and nx.has_path(graph,poss_sense.name(),sense.name()):
           p = nx.shortest_path_length(graph,poss_sense.name(),sense.name())
           sum = sum + math.exp(-(p-1))
return sum
```

Una volta definito il calcolo dell score possiamo calcolare probabilità condizionata di ottenere il senso *s* dal termine *w*:

$$p(s,w) = \frac{score(s,w)}{\sum_{s' \in SensesWN(w), w' \in Ctx(w)} score(s',w')}$$

Scegliamo il valore che restituisce il massimo, ossia:

$$bestSense\ s = argmax\ p(sw)$$

in code:

```
prob = scores[i][1] / denominatore if denominatore != 0 else 0
if prob>=best_prob:
   best_prob=prob
   best_sense=i
```