Relazione esercizio FCA

L'ultima esercitazione ha riguardato la Formal Concept Analisys come metodologia per fare ontology learning. L'Ontology learning rappresenta il processo di creazione automatica di una ontologia a patire dai testi e dai documenti che si hanno a disposizione. Si può anche vedere come il passaggio da un dato non strutturato, come un documento di testo, ad un dato strutturato in maniera automatica. Una delle diverse metodologie che abbiamo studiato per fare ontology learning è la Formal Concept Analisys. Quest'ultima rappresenta un metodo che proviene dall'ambito matematico, che consente di indurre automaticamente delle tassonomie a partire da dati non strutturati. Fondamentale per comprendere il meccanismo di FCA è **la matrice di adiacenza**, una matrice le cui righe sono rappresentate dai concetti e le colonne dalle features. Analizzando uno specifico dominio di interesse, ad esempio quello della frutta, se ne ricavano **concetti e features**, come ad esempio: "mela" e "ha la buccia". Le caratteristiche sono modellate come funzioni a valore booleano che individuano la presenza o l'assenza di una proprietà in ogni concetto. Da questa matrice viene costruito il **contesto formale**, una rappresentazione tassonomica dei concetti e delle features a forma di grafo, i cui nodi sono i **concetti latenti** e gli archi sono relazioni iperonimiche. I concetti della matrice vengono fusi in questi nuovi concetti latenti e ne viene ricavata una struttura tassonomica.

Ai fini di questa esercitazione è stato definito un algoritmo di FCA che a partire da informazioni testuali produce il contesto formale assieme al **lattice**, una rappresentazione grafica riassuntiva della struttura tassonomica precedentemente creata. In particolare l'algoritmo analizza un insieme di frasi di un testo dato in input e ne trae gli alberi a dipendenze. Ai fini di ciò è stata usata la libreria spacy per parsificare singole frasi. Dall'analisi sintattica si selezionano le dipendenze sintattiche associate alle parole di contenuto (sostantivi, nomi propri, verbi, aggettivi, avverbi) e da queste se ne ricava l'insieme delle features per costruire il contesto formale. I concetti, invece sono proprio le parole di contenuto parsificate. L'entry point del programma è il file "**main.py**", seguendo il flusso di esecuzione del main si andrà a spiegare nel dettaglio l'implementazione. Il main riceve in ingresso 2 parametri:

- **DATASET**: il path al file di testo
- **METHOD**: il metodo di analisi, se su una sola frase cercata in modo casuale nelle frasi del documento oppure sull'intero documento. Accetta valori: single, all

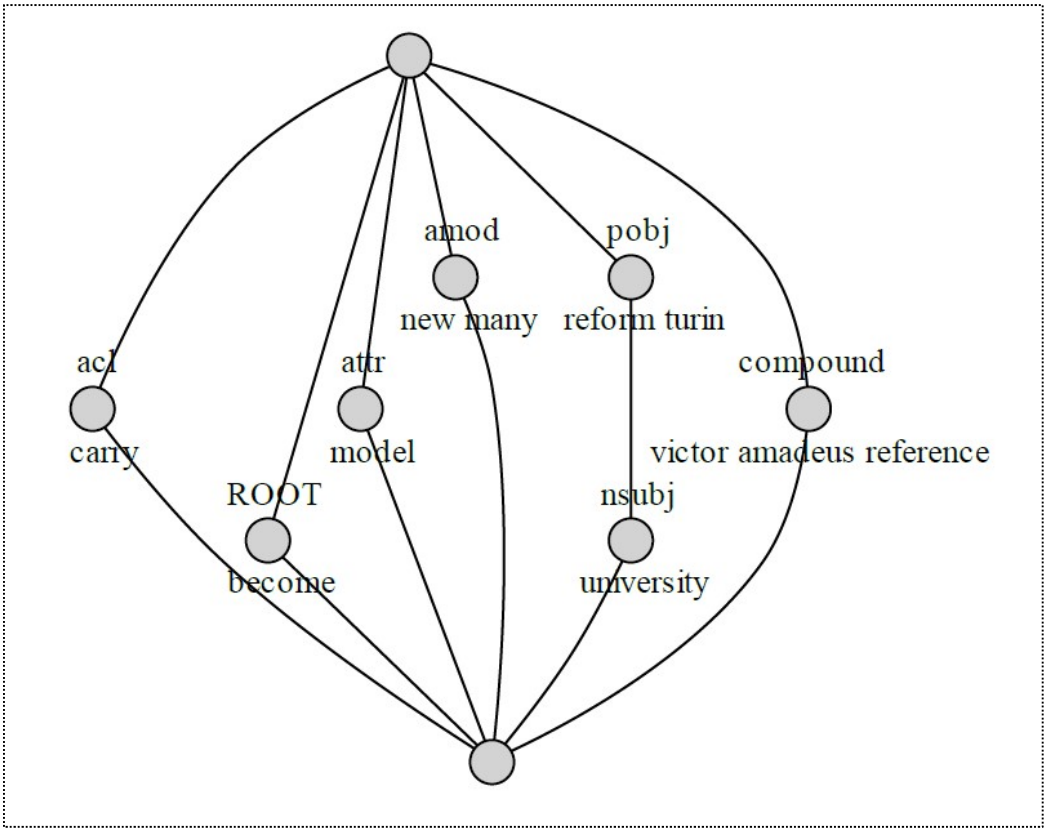
In seguito il documento viene letto e rispettando il metodo di analisi viene parsificato. Di tutte le parole di contenuto analizzate si crea un dizionario, alle quali viene associata la lista delle dipendenze sintattiche trovate per ogni occorrenza della parola nelle frasi. Successivamente viene creata la matrice di adiacenza considerando come features le possibili dipendenze sintattiche trovate e come concetti le parole di contenuto. La matrice così costruita è un file csv che viene salvato nella cartella **"csv"** con lo stesso nome del documento dato in input. Infine usando la libreria "**concepts**" viene costruito il contesto formale chiamando la funzione "**concepts.load_csv**".

Risultati

Dal contesto formale, infine, viene creato il lattice e i risultati vengono riportati a video e salvati come documenti di testo sotto la cartella "**output**". Di seguito si riporta, a titolo di esempio, la matrice di adiacenza, il contesto formale e il lattice ricavato dall'analisi di una frase scelta in modo casuale dal documento "**university_of_turin.txt**" disponibile sotto la cartella **input**, fra i file del progetto. La frase analizzata è la seguente: *"With the reforms carried out by Victor Amadeus II, the University of Turin became a new reference model for many other universities."*

```
<Context object mapping 11 objects to 7 properties [ee7c3d1e] at 0x24fc1adbc70>
             |attr|ROOT|acl|amod|compound|nsubj|pobj|
    reform   |    |    |   |    |        |     |X   |
    carry    |    |    |X  |    |        |     |    |
    victor   |    |    |   |    |X       |     |    |
    amadeus  |    |    |   |    |X       |     |    |
    university|    |    |   |    |        |X    |X   |
    turin    |    |    |   |    |        |     |X   |
    become   |    |X   |   |    |        |     |    |
    new      |    |    |   |X   |        |     |    |
    reference|    |    |   |    |X       |     |    |
    model    |X   |    |   |    |        |     |    |
    many     |    |    |   |X   |        |     |    |
```

```
digraph Lattice {
        node [label="" shape=circle style=filled width=.25]
        edge [dir=none labeldistance=1.5 minlen=2]
        c0
        c1
        c1 -> c1 [color=transparent headlabel=carry labelangle=270]
        c1 -> c1 [color=transparent labelangle=90 taillabel=acl]
        c1 -> c0
        c2
        c2 -> c2 [color=transparent headlabel=university labelangle=270]
        c2 -> c2 [color=transparent labelangle=90 taillabel=nsubj]
        c2 -> c0
        c3
        c3 -> c3 [color=transparent headlabel=become labelangle=270]
        c3 -> c3 [color=transparent labelangle=90 taillabel=ROOT]
        c3 -> c0
        c4
        c4 -> c4 [color=transparent headlabel=model labelangle=270]
        c4 -> c4 [color=transparent labelangle=90 taillabel=attr]
        c4 -> c0
        c5
        c5 -> c5 [color=transparent headlabel="new many" labelangle=270]
        c5 -> c5 [color=transparent labelangle=90 taillabel=amod]
        c5 -> c0
        c6
        c6 -> c6 [color=transparent headlabel="reform turin" labelangle=270]
        c6 -> c6 [color=transparent labelangle=90 taillabel=pobj]
```


amod
pobj
new many
reform turin
acl
attr
compound
carry
model
victor amadeus reference
ROOT
nsubj
become
university