Rapport TP4 RCR

CHIBANE Ilies
181831072041

BOUROUINA Rania
181831052716

1) Introduction:

Dans ce TP nous allons nous intéresser à l'implémentation de réseaux sémantiques, et plus précisément à l'implémentation de différents algorithmes des réseaux sémantiques.

2) Réseaux sémantiques:

Un réseau sémantique est un graphe marqué destiné à la représentation des connaissances. Pour notre TP un réseau sémantique est défini dans un fichier JSON récupéré via le site qui a été joint dans le TP ou chaque nœud possède un label et un id qui sont par la suite utilisés pour représenter les différentes relations entre les nœuds. Nous allons réaliser les différents algorithmes avec le langage python étant particulièrement adapté pour travailler avec les fichiers JSON.

3) Partie 1 : implémenter l'algorithme de propagation de marqueurs dans les réseaux sémantiques:

Dans cette partie nous allons implémenter l'algorithme de propagation de marqueurs vu en cours :

```python
import json

def get_label(reseau_semantique, node, relation):
    node_relation_edges = [edge["from"] for edge in reseau_semantique["edges"] if (edge["to"] == node["id"] and edge["label"] == relation)]
    node_relation_edges_label = [node["label"] for node in reseau_semantique["nodes"] if node["id"] in node_relation_edges]
    reponse = "il y a un lien entre les 2 noeuds : " + ", ".join(node_relation_edges_label)
    return reponse

def propagation_de_marqueurs(reseau_semantique, node1, node2, relation):
    nodes = reseau_semantique["nodes"]

    solutions_found = []

    for i in range(min(len(node1),len(node2))):
        solution_found = False

        try:
            M1 = [node for node in nodes if node["label"] == node1[i]][0]
            M2 = [node for node in nodes if node["label"] == node2[i]][0]

            edges = reseau_semantique["edges"]
            propagation_edges = [edge for edge in edges if (edge["to"] == M1["id"] and edge["label"] == "is a")]
            while len(propagation_edges) != 0 and not solution_found:
                temp_node = propagation_edges.pop()
                temp_node_contient_edges = [edge for edge in edges if (edge["from"] == temp_node["from"] and edge["label"] == relation)]
                solution_found = any(d['to'] == M2["id"] for d in temp_node_contient_edges)
                if not solution_found:
                    temp_node_is_a_edges = [edge for edge in edges if (edge["to"] == temp_node["from"] and edge["label"] == "is a")]
                    propagation_edges.extend(temp_node_is_a_edges)

            solutions_found.append(get_label(reseau_semantique, M2, relation) if solution_found else "il n'y a pas un lien entre les 2 noeuds")
        except IndexError:
            solutions_found.append("Aucune reponse n'est fournie par manque de connaissances.")

    return(solutions_found)
```

Code 1 - Algorithme de propagation de marqueur

Pour cet Algorithme nous allons utiliser le réseau sémantique fournis par le lien donnée dans la série de TP détaillé dans l'image suivante :

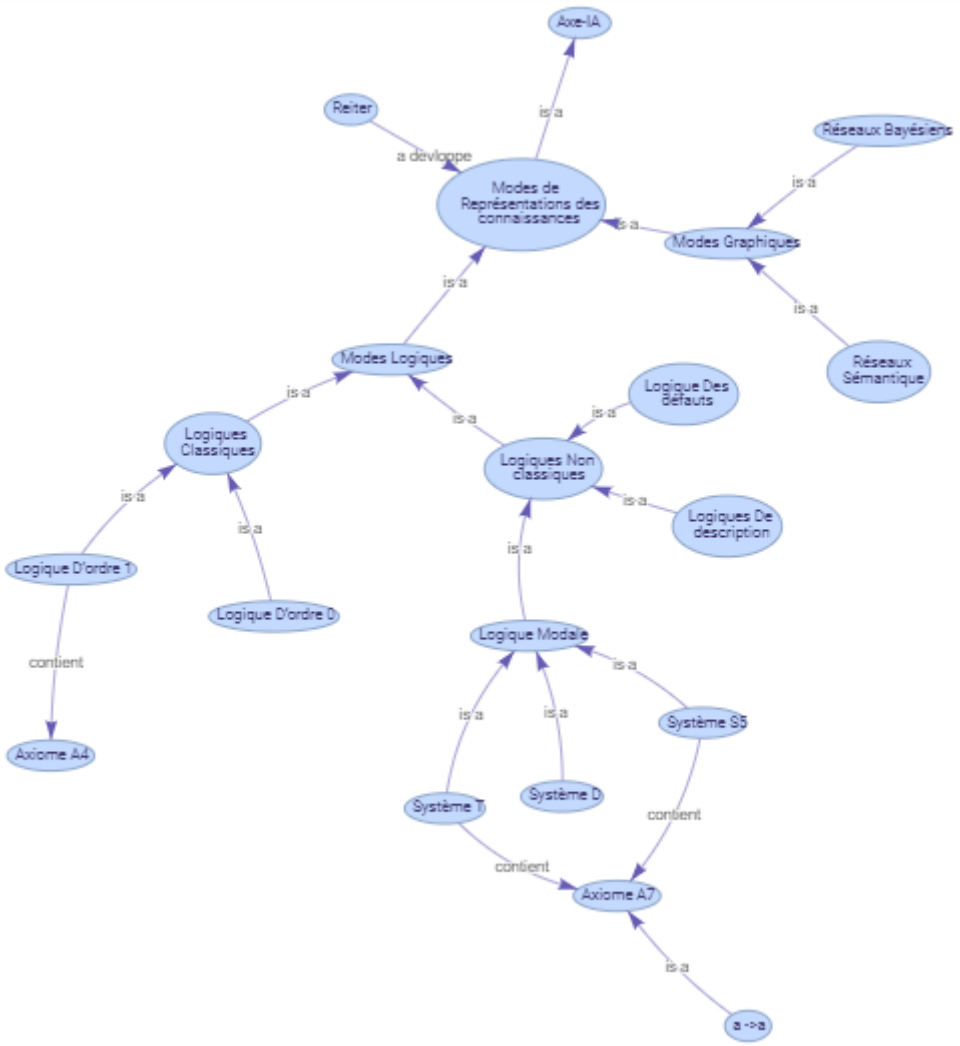

Figure 1 – Réseau sémantique utilisé pour la partie 1 du TP

Utilisant plusieurs noeud marqué en entré on obtient les résultats suivant :

```
Partie 1: l'algorithm de propagation de marqueurs
Modes de Representations des connaissances contient Axiome A7
il y a un lien entre les 2 noeuds : Systeme T, Systeme S5
Modes de Representations des connaissances contient Axiome A4
il y a un lien entre les 2 noeuds : Logique D ordre 1
Modes de Representations des connaissances contient Axe-IA
il n'y a pas un lien entre les 2 noeuds
Modes de Representations des connaissances contient Axiome A9
Aucune reponse n’est fournie par manque de connaissances.
```

Figure 2 – Résultat obtenu par l’algorithme de propagation de marqueurs

4) Partie 2 : implémenter l'algorithme d'héritage:

Dans cette partie nous allons implémenter l'algorithme d'héritage vu en cours:

```python
import json

def get_label(reseau_semantique, node_id):
    label = [node["label"] for node in reseau_semantique["nodes"] if node["id"] == node_id]
    return " ,".join(label)

def heritage(reseau_semantique, name):
    the_end = False

    nodes = reseau_semantique["nodes"]
    edges = reseau_semantique["edges"]

    node = [node for node in nodes if node["label"] == name][0]
    legacy_edges = [edge["to"] for edge in edges if (edge["from"] == node["id"] and edge["label"] == "is_a")]
    legacy = []
    properties = []
    while not the_end:
        n = legacy_edges.pop()
        legacy.append(get_label(reseau_semantique, n))
        legacy_edges.extend([edge["to"] for edge in edges if (edge["from"] == n and edge["label"] == "is_a")])
        properties_nodes = [edge for edge in edges if (edge["from"] == n and edge["label"] != "is_a")]
        for pn in properties_nodes:
            properties.append(": ".join([pn["label"], get_label(reseau_semantique, pn["to"])]))
        if len(legacy_edges) == 0:
            the_end = True

    return legacy, properties
```

Code 2 - Algorithme d'héritage

Pour cet Algorithme nous allons utiliser un autre réseau sémantique exploitant mieux les propriétés de cet algorithme et qui a lui aussi été vu en cours et qui est détaillé dans l'image suivante :

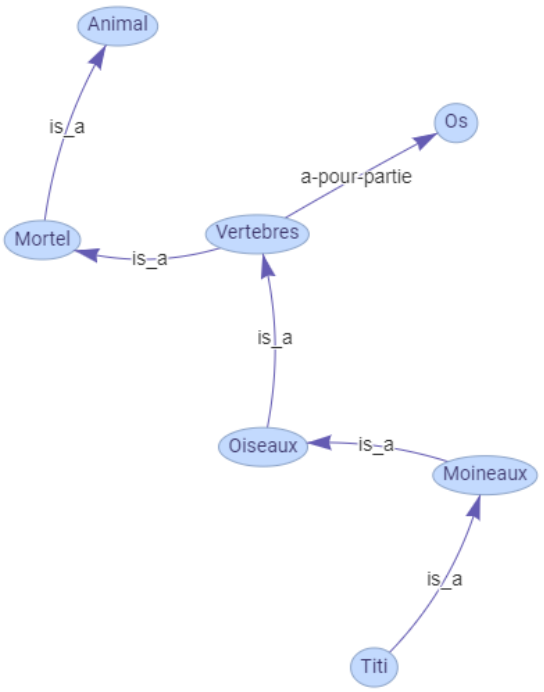

Figure 3 – Réseau sémantique utilisé pour la partie 2 du TP

En utilisant le noeud **“Titi”** en entré on obtient le résultat suivant :

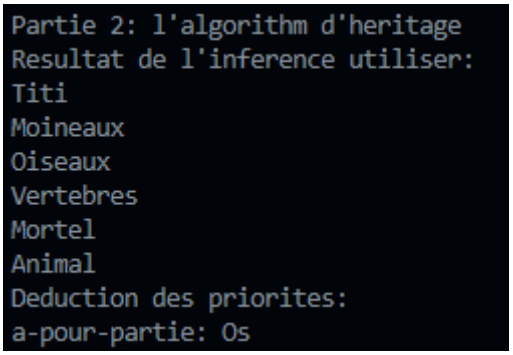

Figure 4 – Résultat obtenu par l’algorithme d’héritage

5) Partie 3 : implémentez un algorithme qui permet d’inhiber la propagation dans le cas des liens d’exception:

Cet algorithme est très similaire à celui de la partie 1 la seule différence est dans le faite qu’on ne prend pas en compte les arcs de types exception l’algorithme est plus détaillé ci-dessous :

```python
import json

def get_label(reseau_semantique, node, relation):
    node_relation_edges = [edge["from"] for edge in reseau_semantique["edges"] if (edge["to"] == node["id"] and edge["label"] == relation)]
    node_relation_edges_label = [node["label"] for node in reseau_semantique["nodes"] if node["id"] in node_relation_edges]
    reponse = "il y a un lien entre les 2 noeuds : " + ", ".join([node_relation_edges_label[1]])
    return reponse

def propagation_de_marqueurs(reseau_semantique, node1, node2, relation):
    nodes = reseau_semantique["nodes"]

    solutions_found = []

    for i in range(min(len(node1),len(node2))):
        solution_found = False

        try:
            M1 = [node for node in nodes if node["label"] == node1[i]][0]
            M2 = [node for node in nodes if node["label"] == node2[i]][0]

            edges = reseau_semantique["edges"]
            propagation_edges = [edge for edge in edges if (edge["to"] == M1["id"] and edge["label"] == "is a" and edge["edge_type"] != "exception")]
            while len(propagation_edges) != 0 and not solution_found:
                temp_node = propagation_edges.pop()
                temp_node_contient_edges = [edge for edge in edges if (edge["from"] == temp_node["from"] and edge["label"] == relation and edge["edge_type"] != "exception")]
                solution_found = any(d['to'] == M2["id"] for d in temp_node_contient_edges)
                if not solution_found:
                    temp_node_is_a_edges = [edge for edge in edges if (edge["to"] == temp_node["from"] and edge["label"] == "is a" and edge["edge_type"] != "exception")]
                    propagation_edges.extend(temp_node_is_a_edges)

            solutions_found.append(get_label(reseau_semantique, M2, relation) if solution_found else "il n'y a pas un lien entre les 2 noeuds")
        except IndexError:
            solutions_found.append("Aucune reponse n’est fournie par manque de connaissances.")

    return(solutions_found)
```

Code 2 - Algorithme d'inhibition de la propagation dans le cas des liens d’exception

Pour cet Algorithme nous allons utiliser le même réseau sémantique avec l’ajout d’un lien d’exception pour voir la différence avec le premier algorithme. Le réseau est détaillé dans l’image suivante :

Figure 5 – Réseau sémantique utilisé pour la partie 3 du TP

Utilisant la question “Mode de représentation des connaissances contient Axiome A7” on obtient la réponse suivante :

```
Partie 3: l'algorithm de propagation de marqueurs avec exception
Modes de Representations des connaissances contient Axiome A7
il y a un lien entre les 2 noeuds : Systeme S5
```

Figure 6 – Résultat obtenu par l’algorithme d'inhibition de la propagation dans le cas des liens d’exception