L’interface:

1. Pour le développement de l’interface web, en premier temps j’essaye de implementer l’algorithme total dans le code javascript, mon idée était: je insère un catégorie dans l’interface, fais les requête sparql et affiche les résultats directement sur l’interface. Mais le requête sparql ne marche pas dans javascript.

Alors je separe les deux parties: analyser et afficher.

Je peux créer un fichier JSON dans lequel enregistre tous les categories qu’on veut étudier, ex:

Category_list.json———

```
[
  {
    "category": "Édifice-type"
  },
  {
    "category": "Programmation_informatique"
  },
  {
    "category": "Droit_des_sociétés"
  },
  {
    "category": "Technique_musicale"
  },
  {
    "category": "Événement_sportif_international"
  },
  {
    "category": "Région_d'Europe"
  }
 …..
]
```

dans le code JAVA, je lis ce fichier Category_list.json et faire l’analyse sur l’un catégorie après l’autre. Les triples obtenus seront sauvegarde dans des fichiers dont les noms sont liées vers le nom du catégorie.

2. dans le code JAVA, il y a deux moyens pour sauvegarder les résultats. le premier est ce que j’ai indiqué dans le compte-rendu précédant, je sépare les résultats qui existent dans le DBpedia et les-quels n’existent pas, par example:

Category_Technique_musicale_existRes.json
Category_Technique_musicale_notExistRes.json

l’autre moyen est que je sauvegarde les triples d’un catégorie dans juste un fichier JSON, c’est plutôt mieux les visualiser dans l’interface web.

Category_Technique_musicale.json

3. Quand je fais l'affichage des résultats sur l'interface web, au debut je les affiche dans un tableau. Mais après quand j'étudie ce library, je trouve un type de diagramme plus intéressant

J'implement le d3.js dans le code javascript. Quand j'étudie ce library, je trouve un type de diagramme plus intéressant que le tableau, c'est le Cluster Dendrogram, comme l'example suivant:

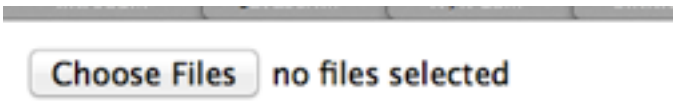
Alors j'essaye d'implementer le d3.js et affiche les résultats en format du Cluster Dendrogram, au lieu du tableau.

4. La description de l'interface

Au debut, on peut sélectionner un fichier locale;

code:

```
<input type="file" id="files" name="files[]" multiple />
<output id="list"></output>
```



si le fichier est bien téléchargé, une alerte va afficher disant ‘suscess’ . on peut aussi voir les informations sur ce fichier.

code:

```
var files = evt.target.files; // FileList object

    // files is a FileList of File objects. List some properties.
    var output = [];
    for (var i = 0, f; f = files[i]; i++) {
      output.push(f.name, '</strong> (', f.type || 'n/a', ') - ',
                  f.size, ' bytes, last modified: ',
                  f.lastModifiedDate ? f.lastModifiedDate.toLocaleDateString() : 'n/a',
                  '</li>');
```

Si on choisi un des fichier JSON qu’on obtenue du code JAVA, les triples vont être affiché dans le page comme un arbre.

Ici, j’utilise le méthode d3.json,la racine(root) soit le catégorie, il a deux branches qui sont ‘existDB’ et ‘notExistDB’, puis tous les triples de ce catégorie vont être liées à les deux branches, par l’ordre sc—ps—os.

Le code :

```
d3.json(str, function(error, root) {
  var nodes = cluster.nodes(root),
      links = cluster.links(nodes);

// les lignes entre des nodes adjacents
  var link = svg.selectAll(".link")
      .data(links)
    .enter().append("path")
      .attr("class", "link")
      .attr("d", diagonal);

  var node = svg.selectAll(".node")
      .data(nodes)
    .enter().append("g")
      .attr("class", "node")
      .attr("transform", function(d) { return "translate(" + d.y + "," + d.x + ")"; })

  node.append("circle")
      .attr("r", 4.5);

  node.append("text")
      .attr("dx", function(d) { return d.children ? -8 : 8; })
      .attr("dy", 3)
      .style("text-anchor", function(d) { return d.children ? "end" : "start"; })
      .text(function(d) { return d.name; });
});
```

5 . L’effet de l’affichage

Je prends le catégorie Technique_musicale comme un example, le résultat est comme suivant: