INSA 5ème année
Alignement d'ontologies

Présentation de l'environnement de TP

Pour ce TP, vous allez générer et manipuler des alignements entre deux ontologies à l'aide de l'*Alignment API* et de différents matcheurs. L'API est disponible sur `http://alignapi.gforge.inria.fr/`. Téléchargez la version 4.7. Plusieurs tutoriels sur l'API sont disponibles sur `http://alignapi.gforge.inria.fr/tutorial/`. Dans le cadre de ce TP, vous allez utiliser l'API au sein d'une application Java [1].

1 Alignment API

1.1 Les ontologies

Vous allez générer plusieurs alignements entre deux ontologies du domaine de la bibliographie [2]. Pour ces deux ontologies, l'alignement de référence est également disponible. Cela vous permettra d'évaluer (en termes de précision, rappel et f-measure) les alignements générés.

- Benchmark 101 : `http://oaei.ontologymatching.org/tests/101/onto.rdf`
- Real ontology INRIA : `http://oaei.ontologymatching.org/tests/304/onto.rdf`
- Alignement de référence : `http://oaei.ontologymatching.org/tests/304/refalign.rdf`

1.2 Les matcheurs simples de l'API

Ci-dessous, vous trouverez le fragment de code nécessaire pour générer un alignement simple via l'API :

```
public static void generateAlign() throws URISyntaxException,
    AlignmentException {
  URI onto1=new URI("http://oaei.ontologymatching.org/tests/101/onto.rdf");
  URI onto2=new URI("http://oaei.ontologymatching.org/tests/302/onto.rdf");

  AlignmentProcess alignment = new NameEqAlignment();
  alignment.init (onto1, onto2);
  alignment.align(null, new Properties());

  System.out.println("Num corresp. générées : " + alignment.nbCells());
}
```

Dans cet exemple, le processus d'alignement est basé sur la comparaison des noms (ID) des entités des deux ontologies, en utilisant une mesure d'égalité entre chaînes de caractères. D'autres matcheurs de base sont disponibles dans API, notamment :

1. Pour vos projets Eclipse, vous devez importer les fichiers lib/procalign.jar, lib/align.jar et lib/ontowrap.jar
2. Notez que ces deux ontologies sont utilisées dans le contexte des campagnes d'évaluations OAEI et sont utilisées ici à titre d'exemple.

— `EditDistNameAlignment` : utilise une distance d'édition (distance de Levenshtein) pour aligner les entités de deux ontologies (cette méthode peut générer une correspondance entre une classe et une propriété) ;
— `SMOANameAlignment` : utilise une distance d'édition [1] pour aligner les entités de deux ontologies ;
— `NameAndPropertyAlignment` : utilise une mesure de substring pour aligner les classes et les propriétés de deux ontologies (cette méthode ne génère pas une correspondance entre une classe et une propriété) ;
— `ClassStructAlignment` : utilise une mesure de substring pour aligner les classes ayant des propriétés en commun.

Ces matcheurs sont des algorithmes de base qui peuvent être utilisé pour le développement d'algorithmes plus élaborés.

Pour écrire l'alignement généré dans un fichier .rdf, vous pouvez utiliser le code ci-dessous :

```
public static void render (Alignment alignment) throws
    FileNotFoundException, UnsupportedEncodingException, AlignmentException
     {
   PrintWriter writer ;
   FileOutputStream f = new FileOutputStream (new File("PATH") ) ;
   writer = new PrintWriter (new BufferedWriter (new OutputStreamWriter (f,
       "UTF-8" )), true) ;
   AlignmentVisitor renderer = new RDFRendererVisitor (writer) ;
   alignment.render(renderer) ;
   writer.flush() ;
   writer.close() ;
}
```

Notez que `AlignmentProcess` extends `Alignment`.

1.3 Évaluation d'alignements

Dans le cas général, l'évaluation d'un alignement généré par un algorithme d'alignement se fait en le comparant à un alignement de référence, créé manuellement par un expert du domaine. Avec l'API, un alignement peut être évalué comme suit :

```
public static void evaluate(Alignment alignment) throws URISyntaxException,
     AlignmentException {
  URI reference = new URI("http://oaei.ontologymatching.org/tests/302/
      refalign.rdf");

  AlignmentParser aparser = new AlignmentParser(0);
  Alignment refalign = aparser.parse(reference);
  PRecEvaluator evaluator = new PRecEvaluator(refalign, alignment);
  evaluator.eval(new Properties());

  System.out.println("Precision : " + evaluator.getPrecision());
  System.out.println("Recall :"     + evaluator.getRecall());
  System.out.println("FMeasure :"   + evaluator.getFmeasure());
}
```

Travail à faire

1. Faites une analyse des deux ontologies données (§1.1) (structures, présence de propriétés, labels, commentaires, etc.). Vous pouvez utiliser Protégé pour cela ;

2. Générez des alignements pour ces deux ontologies, en utilisant quatre matcheurs de votre choix ;
3. Générez les fichiers RDF correspondants ;
4. Faites une analyse de chaque alignement généré, par rapport à la façon dont la technique utilisée exploite les caractéristiques des ontologies alignées ;
5. Comparez les alignements générés à l'aide des mesures de précision, rappel et f-measure. Quelle technique exploite le mieux les caractéristiques des ontologies ?

2 Développement d'un matcheur

Dans cette deuxième partie du TP, vous allez étendre un matcheur de base développé à l'aide de l'Alignment API et qui compare (en utilisant la mesure d'égalité de chaînes de caractères) les labels des concepts de deux ontologies [3]. Ce matcheur est basé sur l'API OWL [4]. Téléchargez la version OWL API (for OWL 2.0) 4.1.0. Un tutoriel détaillant la création d'un matcheur en utilisant l'API est également disponible [5].

Vous devez :

— étudier le code fourni et le modifier afin de prendre en compte les entités des types `object property` et `data property` dans le processus d'alignement ;
— utiliser une mesure de comparaison autre que l'égalité entre chaînes de caractères (par exemple, la distance d'édition fournie en Annexe A).

3 Utilisation du matcheur LogMap

LogMap [2] est un outil d'alignement basé sur la combinaison de plusieurs méthodes d'alignements (la comparaison lexicale entre les labels des entités, le raisonnement sur l'hiérarchie de classes, et le raisonnement et la réparation d'alignements). Cet outil est publiquement disponible [6] sous licence libre. LogMap peut être exécuté en ligne de commande comme suit :

```
java -jar logmap2_standalone.jar MATCHER URI-onto1 URI-onto2 FULL-PATH-
    OUTPUT-FOLDER [true | false]
```

Vous pouvez également appeler LogMap à partir d'une classe Java :

```
String onto1_iri = "PATH";
String onto2_iri = "PATH";
OWLOntologyManager onto_manager = OWLManager.createOWLOntologyManager();

OWLOntology onto1 = onto_manager.loadOntology(IRI.create(onto1_iri));
OWLOntology onto2 = onto_manager.loadOntology(IRI.create(onto2_iri));

LogMap2_Matcher logmap2 = new LogMap2_Matcher(onto1, onto2);
Set<MappingObjectStr> logmap2_mappings = logmap2.getLogmap2_Mappings();
System.out.println("Num. mappings : " + logmap2_mappings.size());
```

Vous devez :

— faire tourner LogMap en utilisant en entrée les ontologies décrites dans §1.1 ;
— comparer les alignement générés par les algorithmes de l'API (§1.2), par le matcheur simple (§2) et LogMap à l'aide des mesures de précision, rappel et f-measure.

3. http://www.irit.fr/~Cassia.Trojahn/NewMatcher.java
4. http://owlapi.sourceforge.net/
5. http://alignapi.gforge.inria.fr/tutorial/tutorial3/index.html
6. http://www.cs.ox.ac.uk/isg/projects/LogMap/

4 Alignement de l'ontologie Cinèma et l'ontologie DBPedia

Dans cette troisième partie du TP, vous utiliserez comme ontologie source, l'ontologie du Cinéma que nous vous fournissons [7]. La base de connaissances est également fournie [8]. L'ontologie cible à être utilisée est l'ontologie DBPedia [9]. Il s'agit d'une ontologie multi-domaine ayant un large éventail d'entités couvrant différents domaines de connaissance (information géographique, personnes, organisations, films, musique, livres, etc.). Cette ontologie est utilisée pour décrire les entités de la base de connaissance DBPedia.

Vous devez :

— faire tourner les algorithmes de l'API (§1.2), le matcheur simple (§2) et LogMap (§3)en utilisant comme entrées les deux ontologies décrites ci-dessous ;

— indiquer comment feriez-vous pour évaluer les alignements générées ?

A Distance d'édition

```
public class LevenshteinDistance {
  private static int minimum(int a, int b, int c) {
    return Math.min(Math.min(a, b), c);
  }

  public static float computeLevenshteinDistance(String str1,String str2) {
    int[][] distance = new int[str1.length() + 1][str2.length() + 1];

    for (int i = 0; i <= str1.length(); i++)
      distance[i][0] = i;
    for (int j = 1; j <= str2.length(); j++)
      distance[0][j] = j;

    for (int i = 1; i <= str1.length(); i++)
      for (int j = 1; j <= str2.length(); j++)
        distance[i][j] = minimum(
            distance[i - 1][j] + 1,
            distance[i][j - 1] + 1,
            distance[i - 1][j - 1]+ ((str1.charAt(i - 1) == str2.charAt(j -
                1)) ? 0 : 1));

    return (float) (1.0-((float)(distance[str1.length()][str2.length()])/
        Math.max(str1.length(), str2.length())));
  }
}
```

Références

[1] G. Stoilos, G. Stamou, and S. Kollias *A String Metric for Ontology Alignment*. International Semantic Web Conference. 2005.

[2] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau. LogMap : Logic-based and Scalable Ontology Matching. In 10th International Semantic Web Confernece (ISWC 2011).

7. http://www.irit.fr/recherches/MELODI/ontologies/FilmographieV1.owl
8. http://www.irit.fr/recherches/MELODI/ontologies/FilmographieV1_Instances.owl
9. http://wiki.dbpedia.org/Ontology (Télécharchez le fichier 'DBPedia Ontology T-Box (Schema)'.)