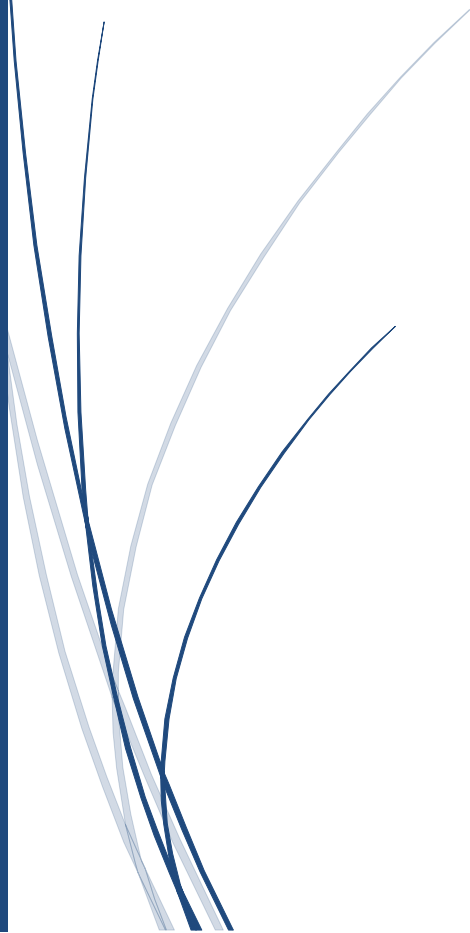




04/03/2022

Interconnexion des données ouvertes liées

Projet web sémantique



Amina FERHATI, Abdellatif MISSOUMI
M2 DATASCALE

Table des matières

Introduction :	2
Interconnexion des données :	2
Méthodes d'interconnexion des données :	2
I LIMES :	3
Définition et fonctionnement :	3
Composants d'un fichier de configuration XML de LIMES :	6
Installation de LIMES :	10
2. SILK :	12
Définition :	12
Sources des données :	12
Règles de liaison :	12
Mesures de similarité :	14
Transformations :	15
Silk workbench :	16
Soutien communautaire :	17
Tableau de synthèse de comparaison :	28
Cas d'utilisation :	29
Conclusion :	33

Introduction :

Dans ce travail, nous allons présenter deux cadres utilisés pour la découverte de liens entre les ensembles de données. Le premier outil est LIMES, il a été publié en 2011, la dernière version a été publiée en 2021, et il s'agit de la 1.3.0 que nous allons comparer au deuxième outil SILK 3.5.0 qui a été publiée en mai 2021. Nous n'avons pas utilisé la dernière version de SILK car nous avons eu des difficultés à l'installer sous Windows.

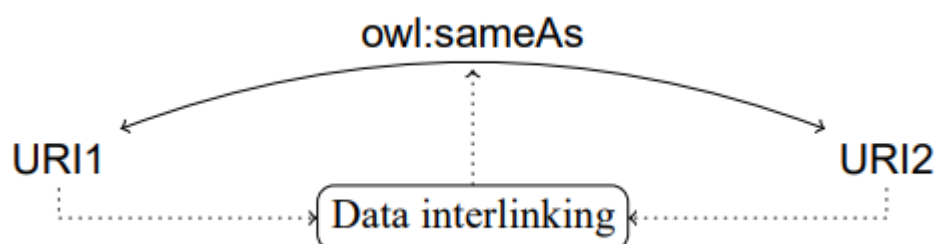
Interconnexion des données :

Le web est constitué de données publiées de telle sorte qu'elles puissent être interprétées et connectées entre elles. Il est donc essentiel d'établir des liens entre ces données.

Le web de données nécessite de relier entre les différentes sources de données publiées. Étant donné la grande quantité de données publiées, il est nécessaire de fournir des moyens de relier automatiquement ces données. De nombreux outils ont été récemment proposés pour résoudre ce problème, chacun ayant ses propres caractéristiques. Dans de nombreux cas, les ensembles de données contenant des ressources similaires sont publiés en utilisant des ontologies différentes. Par conséquent, les outils d'interconnexion de données doivent réconcilier ces ontologies avant de trouver les liens entre les entités. Cette opération peut être réalisée automatiquement, mais le plus souvent, elle est effectuée manuellement et intégrée dans les spécifications des liens.

Méthodes d'interconnexion des données :

Un des principaux problèmes du web de données est de créer des liens entre les entités de différents ensembles de données. Le plus souvent, cela consiste à identifier la même entité dans différents ensembles de données et à publier un lien entre eux sous la forme d'une déclaration owl:sameAs (abrégée en sameAs ci-après). Nous appelons cette tâche l'interconnexion des données et nous la résumons dans la figure ci-dessous.



Une fois les ensembles de liens construits, deux approches sont proposées pour retrouver les équivalences entre les ressources :

- Il est possible d'attribuer à chaque entité du monde réel un identifiant global qui sera ensuite relié à tous les URI décrivant cette entité. C'est l'approche adoptée dans le projet OKKAM (Bouquet et al., 2008).
- L'autre approche utilise des listes d'équivalence maintenues avec des ressources interconnectées à travers des ensembles de données. Il n'y a donc pas d'identifiant global dans cette approche, mais les liens d'équivalence peuvent être suivis à l'aide d'un service Web tiers, par exemple <http://sameas.org>, ou d'un protocole bilatéral (Volz et al., 2009).

L'interconnexion des ensembles de données devient un problème d'autant plus important que leur nombre augmente rapidement. Afin de pouvoir évoluer, la tâche d'interconnexion doit être aussi automatisée que possible. Ces outils d'interconnexion automatiques prennent en entrée deux ensembles de données et fournissent finalement un jeu de liens. En outre, ils utilisent ce que nous appelons une spécification de liaison, c'est-à-dire un "script" spécifiant comment et/ou quoi lier. En effet, compte tenu de la taille des ensembles de données, l'espace de recherche pour l'interconnexion des ressources peut atteindre plusieurs milliards de ressources. Il est donc nécessaire d'utiliser des heuristiques donnant des indications au système d'interconnexion sur l'endroit où chercher les ressources correspondantes dans les deux ensembles de données.

I LIMES :

1 Définition et fonctionnement :

LIMES (Link Discovery Framework for Metric Spaces), est un framework pour découvrir des liens entre des entités contenues dans des sources de données liées. **LIMES** est un framework hybride qui combine les caractéristiques mathématiques des espaces métriques ainsi que le filtrage par préfixe, suffixe et position pour calculer des approximations pessimistes de la similarité des instances. Ces approximations sont ensuite utilisées pour filtrer une grande partie des paires d'instances qui ne remplissent pas les conditions de correspondance. Grâce à ces moyens, LIMES peut réduire le nombre de comparaisons nécessaires au cours du processus de mise en correspondance de plusieurs ordres de grandeur et de complexité sans perdre un seul lien.

Le cadre LIMES se compose de huit modules :

- **Controller** : C'est le module central et il est responsable de la gestion du processus de correspondance.

- **Configuration** : Il communique avec le contrôleur dans le processus de correspondance. Ce dernier commence lorsque le contrôleur appelle le module de configuration, qui lit le fichier de configuration et extrait toutes les informations nécessaires pour effectuer la comparaison des instances, y compris l'URL des points de terminaison SPARQL des bases de connaissances source (S) et cible (T), les restrictions sur les instances à mettre en correspondance, l'expression de la métrique à utiliser et le seuil à utiliser.
- **Query** : Ce module utilise la configuration des bases de connaissances source et cible pour récupérer les instances et les propriétés des points de terminaison SPARQL des bases de connaissances source et cible qui respectent les restrictions spécifiées dans le fichier de configuration. Le module de requête écrit sa sortie dans un fichier en invoquant le module de cache.
- **Rewriter, planner and engine** : Appelé par le contrôleur afin de planifier et d'exécuter la spécification de lien (LS) incluse dans le fichier de configuration, pour identifier l'ensemble des liens (mapping) qui satisfont aux conditions opposées par la LS d'entrée.
- **Machine learning** : Appelé par LIME afin d'exécuter l'algorithme d'apprentissage automatique inclus dans le fichier de configuration, pour identifier un LS approprié pour relier l'ensemble de données source et l'ensemble de données cible. Il procède ensuite à l'exécution du LS.

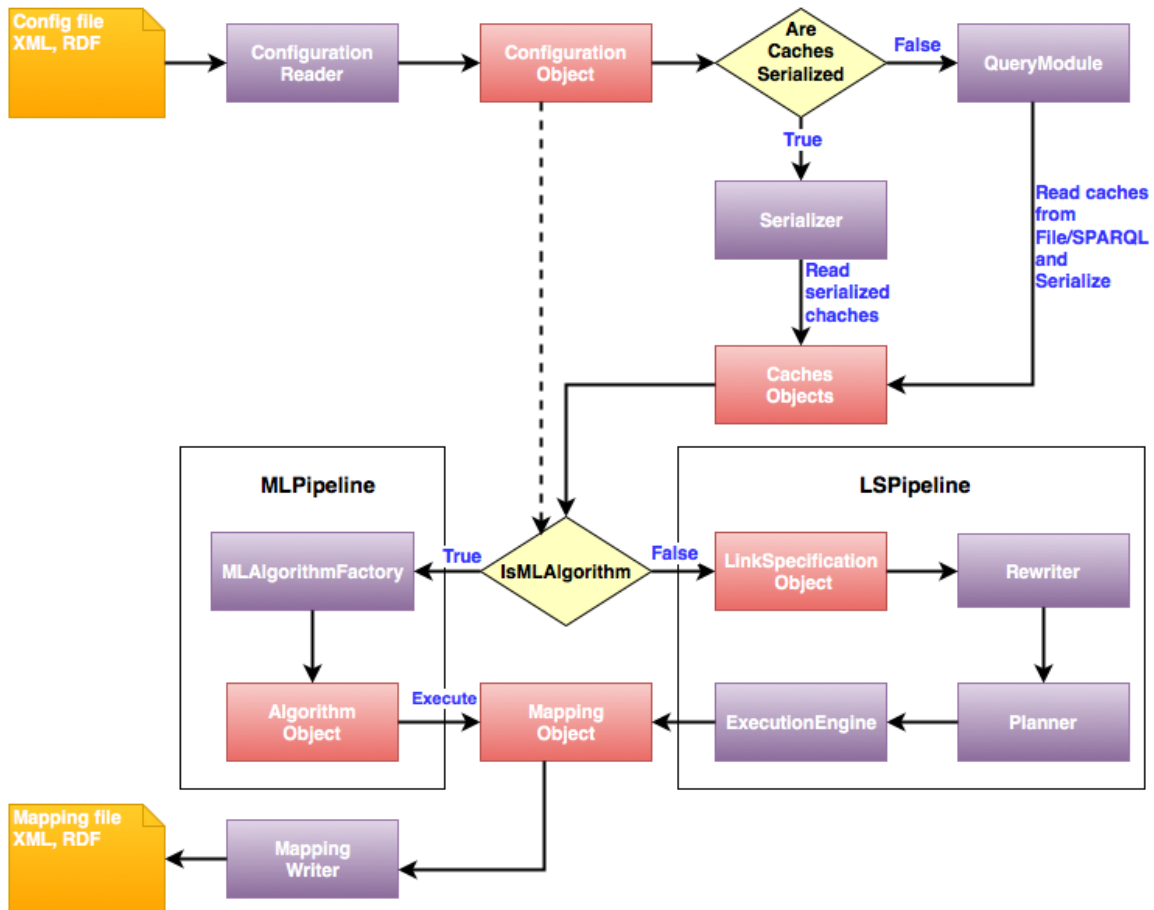


Figure : Les modules composant le framework LIMEs.

LIMES choisit entre la découverte de liens et l'apprentissage automatique. pour les deux tâches, le mappage sera stocké dans le fichier de sortie choisi par l'utilisateur dans le fichier de configuration. Les résultats sont finalement stockés dans un fichier RDF ou XML.

Les avantages du framework LIMES sont :

- Il implémente des mappers hautement optimisés en termes de temps, ce qui le rend une classe de complexité plus rapide que les autres Frameworks de découverte de liens. Ainsi, plus le problème est important, plus LIMES est rapide par rapport aux autres Frameworks de découverte de liens.
- Prend en charge un vaste ensemble de mesures de similarité de type chaîne de caractères, numérique, topologique et temporel, qui permettent à l'utilisateur d'effectuer diverses comparaisons entre les ressources.
- Est garanti pour conduire exactement à la même correspondance qu'une approche par force brute, tout en réduisant considérablement le nombre de comparaisons.
- Supporte un grand nombre de formats d'entrée et de sortie et peut facilement être étendu avec des algorithmes personnalisés, des types de données, des fonctions de prétraitement et plus encore grâce à son architecture modulaire.

2. Composants d'un fichier de configuration XML de LIMES :

Pour que LIMES fonctionne, nous devons générer un fichier de configuration, ce dernier doit comporter les balises suivantes :

2.1 Metadata : se compose toujours des éléments XML suivants :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LIMES SYSTEM "limes.dtd">
<LIMES>
```

2.2 Data Sources : Les deux ensembles de données sont décrits à l'aide des balises suivantes :

- **ID** : l'id du dataset.
- **ENDPOINT** : il peut être un chemin absolu vers le fichier contenant les données ou l'URI du point d'accès SPARQL correspondant.
- **VAR** : décrit la variable associée à l'extrémité susmentionnée.
- **PAGESIZE** : cette propriété doit être définie comme le nombre maximal de triples retournés par le point de terminaison SPARQL.
- **RESTRICTION** : Ce tag permet de contraindre les entrées qui sont récupérées par le module d'interrogation de LIMES.
- **PROPERTY** : permet de spécifier les propriétés qui seront utilisées lors de la liaison, cette balise de propriété peut également être utilisée pour spécifier le prétraitement des données d'entrée.
- **TYPE**

Dans la figure ci-dessous, nous pouvons voir un exemple d'implémentation des sources de données.

```
<SOURCE>
  <ID>mesh</ID>
  <ENDPOINT>http://mesh.bio2rdf.org/sparql</ENDPOINT>
  <VAR>?y</VAR>
  <PAGESIZE>5000</PAGESIZE>
  <RESTRICTION>?y rdf:type meshr:Concept</RESTRICTION>
  <PROPERTY>dc:title</PROPERTY>
  <TYPE>sparql</TYPE>
</SOURCE>
<TARGET>
  <ID>linkedct</ID>
  <ENDPOINT>http://data.linkedct.org/sparql</ENDPOINT>
  <VAR>?x</VAR>
  <PAGESIZE>5000</PAGESIZE>
  <RESTRICTION>?x rdf:type linkedct:condition</RESTRICTION>
  <PROPERTY>linkedct:condition_name</PROPERTY>
</TARGET>
```

2.2.1 Fonctions de prétraitement : Il existe deux catégories de fonctions de prétraitement dans LIMES simple et complexe :

2.2.1.1 Les fonctions simples : LIMES supporte l'ensemble des fonctions de prétraitement suivantes :

- **nolang** : pour supprimer les balises de langue.
- **lowercase** : pour convertir la chaîne d'entrée en minuscule.
- **uppercase** : pour convertir la chaîne d'entrée en majuscules.
- **number** : pour s'assurer que seuls les caractères numériques, "." et "," sont contenus dans la chaîne d'entrée.
- **replace(String a,String b)** : pour remplacer chaque occurrence de a par b.
- **regexreplace(String x,String b)** pour remplacer chaque occurrence de l'expression régulière x par b.
- **cleaniri** : pour supprimer tous les préfixes des IRIs.
- **celsius** : pour convertir Fahrenheit en Celsius.
- **fahrenheit** : pour convertir Celsius en Fahrenheit.
- **removebraces** : pour supprimer les accolades.
- **regularAlphabet** : pour supprimer les caractères non alphanumériques.
- **uriasstring** : renvoie la dernière partie d'un URI sous forme de chaîne.

2.2.1.2 Les fonctions complexes :

Il est également possible d'utiliser des fonctions de prétraitement complexes, c'est-à-dire des fonctions qui manipulent plusieurs valeurs à la fois. Les fonctions de prétraitement complexes suivantes sont disponibles dans LIMES :

- **concat(property1, property2, glue=",") RENAME newprop** : ceci concatène les valeurs de property1 et property2 dans newprop en utilisant une virgule comme colle entre les valeurs. L'utilisation de l'argument glue= est facultative.
- **split(propriété, splitChar=",") RENAME prop1,prop2** : ceci sépare les valeurs de propriété sur le caractère virgule, la première partie est stockée dans prop1 le reste dans prop2. L'argument splitChar= est obligatoire.
- **toWktPoint(propriété1, propriété2) RENAME wktPoint** : ceci prend les valeurs de propriété1 et propriété2 et les stocke comme POINT(p1 p2). Les valeurs des deux propriétés doivent être des nombres.

2.2.2 Source types : Un type de source peut être défini via <TYPE>. Le type par défaut est défini sur SPARQL mais LIMES supporte également la lecture de fichiers directement depuis les fichiers locaux. Les formats de fichiers supportés sont : **CSV**, **N3**, **N-TRIPLE** et **TURTLE**.

2.3 Préfixes : La définition d'un préfixe dans un fichier LIMES exige de définir deux valeurs : L'espace de nom qui sera adressé par l'étiquette du préfixe.

```
<PREFIX>
  <NAMESPACE>http://www.w3.org/1999/02/22-rdf-syntax-ns#</NAMESPACE>
  <LABEL>rdf</LABEL>
</PREFIX>
```

2.4 Expression de la métrique pour la mesure de la similarité : nous devons spécifier la balise métrique dans le fichier de configuration, pour cela limes offre de nombreuses opérations afin de définir des métriques simples et complexes :

```
<METRIC>trigrams(x.rdfs:label,y.dc:title)</METRIC>
```

- **Opérations métriques** : Les opérations métriques permettent de combiner des valeurs métriques. Elles comprennent les opérateurs MIN, MAX et ADD.
- **Opérations booléennes** : Les opérations booléennes permettent de combiner et de filtrer les résultats des opérations métriques et incluent AND, OR, MINUS.

2.5 Machine learning (optionnelle) : Dans la plupart des cas, trouver une bonne spécification de lien n'est pas une tâche triviale. C'est pourquoi limes a implémenté un certain nombre d'algorithmes d'apprentissage automatique pour la génération automatique de spécifications de liens. Afin d'utiliser un algorithme d'apprentissage automatique dans le fichier de configuration, nous devons mettre le tag MLALGORITHM au lieu du tag METRIC. Par exemple :

```
<MLALGORITHM>
  <NAME>wombat simple</NAME>
  <TYPE>supervised batch</TYPE>
  <TRAINING>trainingData.nt</TRAINING>
  <PARAMETER>
    <NAME>max execution time in minutes</NAME>
    <VALUE>60</VALUE>
  </PARAMETER>
</MLALGORITHM>
```

2.6 Condition d'acceptation : Remplir la condition d'acceptation consiste à fixer la valeur seuil à la valeur minimale que deux instances doivent avoir pour satisfaire une relation. Cette opération peut être effectuée comme indiquée ci-dessous :

```

<ACCEPTANCE>
  <THRESHOLD>0.98</THRESHOLD>
  <FILE>accepted.nt</FILE>
  <RELATION>owl:sameAs</RELATION>
</ACCEPTANCE>

```

En utilisant la balise THRESHOLD, nous pouvons définir la valeur minimale que deux instances doivent avoir pour satisfaire la relation spécifiée dans la balise RELATION, c'est-à-dire owl:sameAs dans l'exemple ci-dessus. La balise <FILE> permet de spécifier où les liens doivent être écrits.

2.7 Condition de révision : La définition de la condition à laquelle les liens doivent être examinés manuellement est très similaire à la définition de la condition d'acceptation, comme indiqué ci-dessous :

```

<REVIEW>
  <THRESHOLD>0.95</THRESHOLD>
  <FILE>reviewme.nt</FILE>
  <RELATION>owl:sameAs</RELATION>
</REVIEW>

```

Toutes les instances qui ont une similarité entre le seuil défini dans REVIEW et le seuil défini dans ACCEPTANCE seront écrites dans le fichier de révision et liées via la relation définie dans REVIEW.

2.8 Format de sortie : La cartographie résultante est écrite dans un fichier sur le disque. On peut choisir entre les options suivantes comme format de sortie :

- TAB
- CSV
- NT
- TTL

```

<OUTPUT>NT</OUTPUT>

```

3. Installation de LIMES :

Il y a deux façons d'installer LIMES :

- La première est l'utilisation de docker, afin de l'installer en utilisant docker nous devons exécuter les deux instructions suivantes :

```
docker run -it --rm -v $(pwd):/data dicegroup/limes:latest /data/my-configuration.xml
```

```
docker run -it --rm -p 8080:8080 dicegroup/limes:latest -s
```

Après cela, nous ouvrons le navigateur en utilisant cette URL :

<http://localhost:8080/>

- La deuxième méthode consiste à exécuter ces deux instructions :

```
mvn clean package shade:shade -Dmaven.test.skip=true
```

```
java -jar limes-core/target/limes-core-${current-version}.jar
```

Une fois que nous ouvrons le navigateur sur l'URL **<http://localhost:8080/>**, nous pouvons trouver le LIMES en cours d'exécution dans la page, il a une interface utilisateur graphique ressembler à ceci :

07/02/2022 22:00

LIMES Web UI

TRY EXAMPLES

Select the configuration file

LIMES Web UI

More details about the project

Prefixes

owl uri rdf dbpo

Label

Namespace

ADD

Data source

Sparql endpoint

Local file

Endpoint

http://dbpedia.org/sparql

Restriction

?s rdf:type urt:M

Select restriction

Data target

Sparql endpoint

Local file

Endpoint

http://dbpedia.org/sparql

Restriction

?t rdf:type dbpo:

Select restriction

Manual metric and machine learning

MANUAL METRIC

MACHINE LEARNING

Source property PP

Target property PP

Optional source property PP

Optional target property PP

Start

Operator AND

07/02/2022 22:00

LIMES Web UI

Cosine Threshold 0.5

Source property

Target property

Operator AND

Select Preprocessing Function

Rename As X

Concat Rename A

Metrics

AND(cosine(s.rdfs:label,t.rdfs:label),exactmatch(s.rdfs:label,t.rdfs:label))

EXPORT WORKSPACE TO XML

Acceptance Condition

Threshold 0.98

File accepted.nt

Relation owl:sameAs

Review Condition

Threshold 0.9

File reviewme.nt

Relation owl:sameAs

Output

CSV

11

2. SILK :

2.1. Définition :

Silk est un framework open source qui permet d'intégrer l'intégration des sources de données hétérogènes. Les principaux cas d'utilisation de Silk sont les suivants :

- Générer des liens entre des éléments de données connexes dans différentes sources de données liées.
- Les éditeurs de données liées peuvent utiliser Silk pour établir des liens RDF entre leurs sources de données et d'autres sources de données sur le Web.
- Appliquer des transformations de données à des sources de données structurées.

Silk s'appuie sur le paradigme des données liées, qui repose sur deux idées simples :

- Premièrement, RDF fournit un modèle de données expressif pour représenter des informations structurées.
- Deuxièmement, les liens RDF sont établis entre les entités de différentes sources de données.

Sources des données :

Voici les types de fichiers supportés par Silk :

- Sparql Endpoints
- RDF files
- JSON files
- XML files
- CSV files
- MySQL database

Règles de liaison :

Les règles de liaison sont représentées sous forme d'arbres, qui sont construits à partir de quatre types d'opérateurs :

- **Opérateur de chemin :**

Récupère toutes les valeurs d'un chemin de propriété spécifique de chaque entité, comme la propriété de son étiquette. Le but de l'opérateur de propriété est de permettre l'accès à des valeurs de l'ensemble de données qui sont utilisées comme entrée pour d'autres opérateurs. Chaque déclaration de chemin commence par une variable (telle que définie dans les ensembles de données), qui peut être suivie d'une série d'éléments de chemin. Si un chemin ne peut être résolu en raison d'une propriété manquante ou d'un filtre trop restrictif, un ensemble de résultats vide est renvoyé.

- Pour sélectionner le label en anglais du film en XML :

```
<Input path="?movie/rdfs:label[@lang = 'en']" />
```

- Pour sélectionner le label en anglais du film avec l'API scala :

```
Path.parse("?movie/rdfs:label[@lang = 'en']")
```

- **Opérateur de transformation :**

Transforme les valeurs d'un ensemble d'opérateurs de propriété ou de transformation selon une fonction spécifique de transformation des données. Les exemples de fonctions de transformation comprennent la normalisation de la casse, la tokenisation et la concaténation des valeurs provenant de plusieurs opérateurs. Plusieurs opérateurs de transformation peuvent être imbriqués afin d'appliquer une chaîne de transformations. Ces opérations seront détaillées dans le chapitre "Transformations".

- **Opérateur de comparaison :**

Evalue la similarité entre deux entités sur la base des valeurs renvoyées par deux opérateurs de propriété ou de transformation en appliquant une mesure de distance et un seuil de distance. Des exemples de mesures de distance comprennent la distance de Jaccard ou géographique. Ces mesures de comparaison seront détaillées dans la section "Mesures de similarité".

Voici un exemple avec XML sur la comparaison de deux entités en utilisant la distance de Jaccard après avoir appliqué la transformation de tokenization sur les entités :

```
<Compare metric="jaccard" threshold="0.2">
  <TransformInput function="tokenize">
    <Input path="?a/rdfs:label" />
  </TransformInput>
  <TransformInput function="tokenize">
    <Input path="?b/gn:name" />
  </TransformInput>
</Compare>
```

- **Opérateur d'agrégation :**

Étant donné que, dans la plupart des cas, la similarité de deux entités ne peut pas être déterminée en évaluant une seule comparaison, un opérateur d'agrégation combine les scores de similarité de plusieurs opérateurs de comparaison ou d'agrégation en un seul score selon une fonction d'agrégation spécifique. Parmi les exemples de fonctions d'agrégation courantes, citons la moyenne pondérée ou l'obtention du score minimum de tous les opérateurs.

Une agrégation combine plusieurs valeurs de confiance en une seule valeur. Afin de déterminer si deux entités sont des doublons, il n'est généralement pas suffisant de comparer une seule propriété. Par exemple, lors de la comparaison d'entités géographiques, une agrégation peut regrouper les similarités entre les noms des entités et les similarités basées sur la distance entre les entités.

Voici un exemple en XML d'agrégation de type moyenne, AverageAggregator (average) évalue en utilisant la moyenne (pondérée) des valeurs de confiance :

```
<Aggregate type="average">
  <Compare metric="jaro" required="true">
```

```

    <Input path="?a/rdfs:label" />
    <Input path="?b/gn:name" />
  </Compare>
  <Compare metric="num">
    <Input path="?a/dbpedia:populationTotal" />
    <Input path="?b/gn:population" />
  </Compare>
</Aggregate>

```

Mesures de similarité :

Les mesures de similarité suivantes sont incluses :

- Mesures basées sur les chaînes de caractères :

Fonction et paramètres	Nom	Description
Jaro()	Distance de Jaro	Similitude des chaînes de caractères basée sur la métrique de la distance de Jaro.
jaroWinkler()	Distance de Jaro-Winkler	Similitude des chaînes de caractères basée sur la mesure de distance Jaro-Winkler.
substring([granularity: String = '3'])	Sous-chaîne	Renvoyer 0 à 1 pour une forte similarité à une faible similarité

- Mesures basées sur les tokens :

Si les mesures de distance basées sur les caractères fonctionnent bien pour les erreurs typographiques, il existe un certain nombre de tâches pour lesquelles les mesures de distance basées sur les tokens sont mieux adaptées :

- Les chaînes de caractères dont les parties sont réordonnées, par exemple "John Doe" et "Doe, John".
- Textes composés de plusieurs mots

Fonction et paramètres	Nom	Description
cosine([k: Int = '3'])	cosinus	Mesure de la distance en cosinus.

dice()	Coefficient de dés	Coefficient de similarité de Dice
jaccard()	Jaccard	Coefficient de similarité de Jaccard.

- Mesures basées sur les données numériques :

Fonction et paramètres	Nom	Description
date()	Date	La distance en jours entre deux dates (format 'AAAA MM-JJ').
dateTime()	DateTime	Distance entre deux valeurs de date et d'heure (format xsd:dateTime) en secondes.
insideNumericInterval([separator: String])	Intervalle numérique intérieur	Vérifie si un nombre est contenu dans un intervalle numérique, tel que '1900 - 2000'.

- Mesures basées sur l'égalité :

Fonction et paramètres	Nom	Description
constant([value: Double = '1.0'])	Constante	Renvoie toujours une valeur de similarité constante.
equality()	Egalité	Retourne 0 si les chaînes sont égales, 1 sinon.
insideNumericInterval([separator: String])	Intervalle numérique intérieur	Vérifie si un nombre est contenu dans un intervalle numérique, tel que '1900 - 2000'.

- Il y a d'autres mesures basées sur les relations temporelles et distances ainsi que les relations spatiales.

Transformations :

Les fonctions de transformation et de normalisation suivantes sont incluses :

- Remplacements :

Fonction et paramètres	Nom	Description
replace(search: String, replace: String)	Remplacer	Remplacer toutes les occurrences d'une chaîne de caractères "search" par "replace" dans une chaîne de caractères.

- Normalisation :

Fonction et paramètres	Nom	Description
alphaReduce()	Alpha réduit	Supprime tous les caractères non alphabétiques d'une chaîne de caractères.
lowerCase()	Minuscule	Convertit une chaîne de caractères en minuscules.
removeBlanks()	Supprimer les vides	Supprime les espaces blancs d'une chaîne de caractères.

- Combiner les chaînes de caractères :

Fonction et paramètres	Nom	Description
alphaReduce()	Alpha réduit	Supprime tous les caractères non alphabétiques d'une chaîne de caractères.
lowerCase()	Minuscule	Convertit une chaîne de caractères en minuscules.
removeBlanks()	Supprimer les vides	Supprime les espaces blancs d'une chaîne de caractères.

- Il y a d'autres transformations qui se font sur les dates, les valeurs numériques, les tokenizations, etc.

Silk workbench :

Le Workbench guide l'utilisateur dans le processus de création de tâches de liaison pour interconnecter deux sources de données. Il fournit les composants suivants :

- Navigateur d'espace de travail :

Permet à l'utilisateur de parcourir les projets de l'espace de travail. Les tâches de liaison peuvent être chargées à partir d'un projet et y revenir ultérieurement.

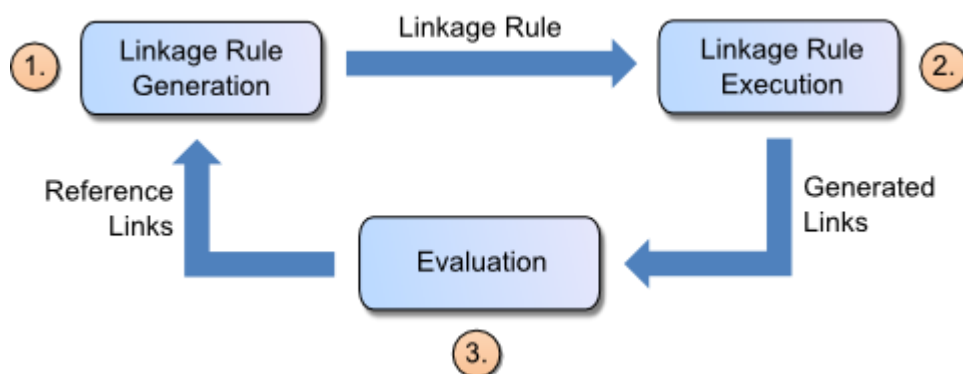
- Éditeur de règles de liaison :

Un éditeur graphique qui permet à l'utilisateur de créer et de modifier facilement des règles de liaison.

- Évaluation :

Permet à l'utilisateur d'exécuter la règle de liaison actuelle. Les liens sont affichés pendant qu'ils sont générés à la volée. L'utilisateur peut demander des résumés détaillés sur la façon dont est composé le score de similarité de liens spécifiques.

Le flux de travail typique pour créer une nouvelle tâche de liaison consiste à :



1. Avant d'exécuter la mise en correspondance proprement dite, il faut construire une règle de liaison, qui spécifie comment deux entités sont comparées pour l'équivalence.
2. La règle de liaison est exécutée, ce qui donne lieu à un ensemble de liaisons.
3. L'objectif de l'étape d'évaluation est de mesurer le succès de la tâche de mise en correspondance des entités et de trouver des erreurs potentielles dans les liens générés.

Soutien communautaire :

Il y a une documentation sur Silk qui est disponible sur GitHub (<https://github.com/silk-framework/silk/tree/master/doc>). Pour les questions et les commentaires, il existe un groupe google (<https://groups.google.com/g/silk-discussion>).

Installation de SILK sous windows :

voici les étapes majeures pour l'installation de cet outil :

- Après avoir lu la documentation d'installation (<https://github.com/silk-framework/silk>), on s'aperçoit que la meilleure façon pour installer cet outil est de passer par le docker.
- Il faut s'assurer tout d'abord que vous posséder JDK8, simple build tool comme sbt et docker (version >=17.05-xx).
- Télécharger le .zip de silk (<https://github.com/silk-framework/silk/releases>), dans notre cas, on a choisi la version 3.5.0
- Lancer l'invite de commande à l'intérieur du dossier silk-v-3.5
- Construire l'image docker avec :

```
sbt universal:packageZipTarball
```

```
docker build -t silkframework/silk-workbench:latest .
```

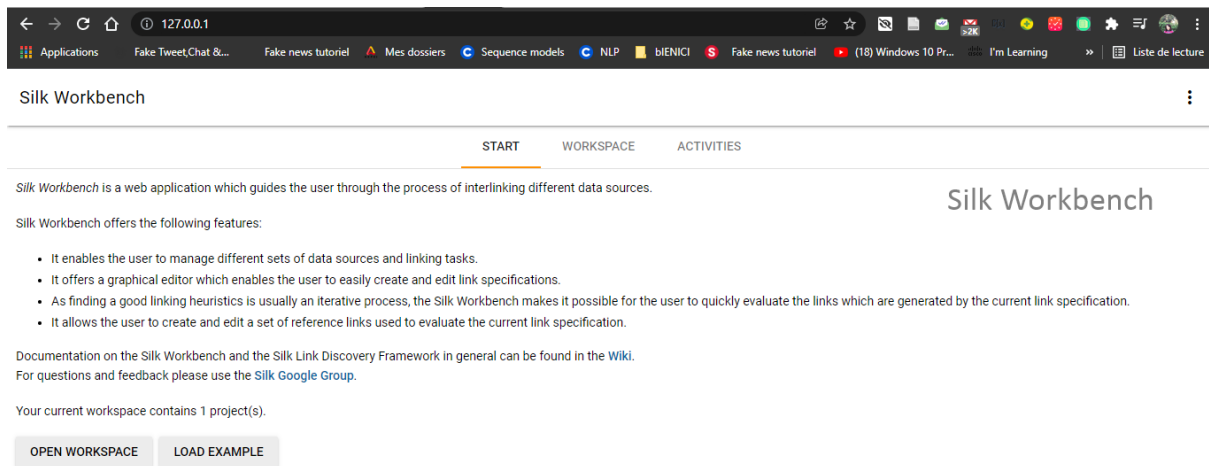
- Pull l'image du docker :

```
docker pull silkframework/silk-workbench
```

- Exécuter le conteneur docker :

```
docker run -d --name silk-workbench -p 80:80 silkframework/silk-workbench:latest
```

- Sur votre navigateur, mettez 127.0.0.1:80 et le workbench de silk sera affiché :



3. Comparaison des outils :

3.1 Évaluation du temps d'exécution :

Une évaluation a été réalisée afin de comparer les performances de LIMES et SILK sur des données réelles. Pour garantir une comparaison objective des temps d'exécution, ils n'ont pris en compte que le temps nécessaire aux deux cadres pour effectuer les comparaisons dans l'évaluation. Chacune des mesures de temps a été effectuée trois fois et seul le meilleur temps d'exécution a été considéré. Ils ont remarqué qu'il n'y avait pas de différence significative entre les différents temps d'exécution de LIMES.

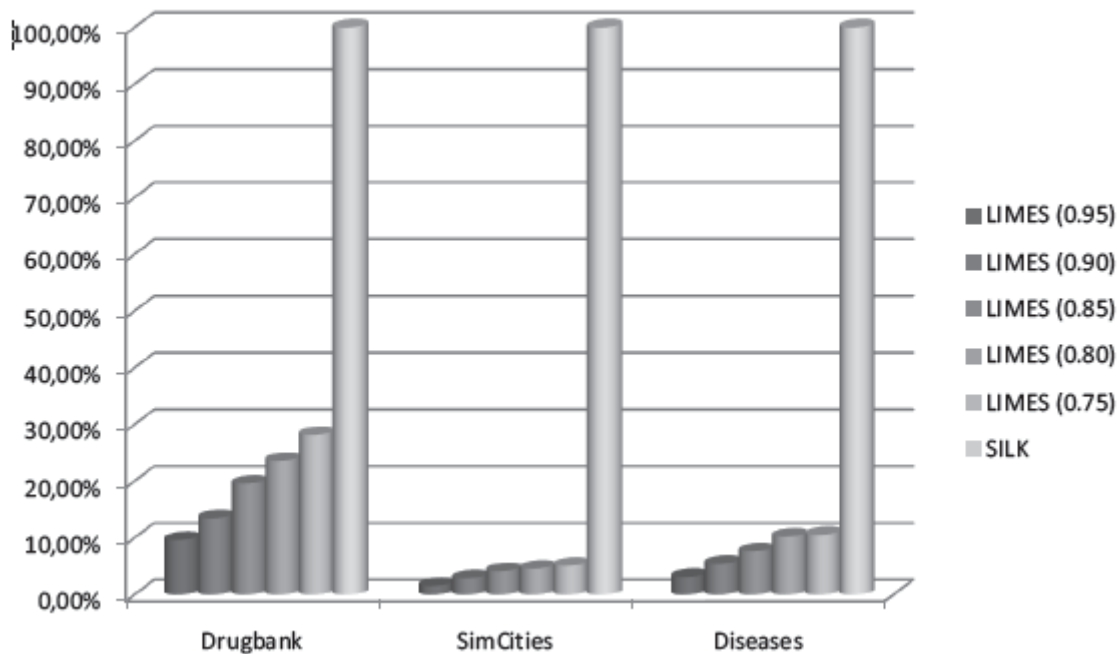
Les expériences sur des données réelles ont été réalisées dans trois contextes différents. L'objectif de la première expérience, appelée **Drugs**, était de cartographier les médicaments dans **DBpedia3** et **Drugbank4** en comparant leurs étiquettes. Le but de la deuxième expérience, nommée **SimCities**, était de détecter les villes dupliquées dans **DBpedia** en comparant leurs étiquettes. Le but de la dernière expérience, nommée **Diseases**, était de cartographier les maladies de MESH5 avec les maladies correspondantes dans LinkedCT6 en comparant leurs étiquettes. Le détail des trois expressions est présenté dans le tableau ci-dessous :

	Drugs	SimCities	Diseases
S	4346	12701	23618
T	4772	12701	5000
E	69	112	70
Source	DBpedia	DBpedia	MESH
Target	Drugbank	DBpedia	LinkedCT

Aperçu des expériences d'exécution. |S| est la taille de la base source, |T| est la taille de la base de connaissances cible et |E| est le nombre d'exemplaires utilisés par LIMES pendant l'expérience. Les résultats de l'expérience sont présentés dans le tableau ci-dessous :

	LIMES					SILK
	0.95	0.9	0.85	0.8	0.75	
Drugs	86	120	175	211	252	1732
SimCities	523	979	1403	1547	1722	33786
Diseases	546	949	1327	1784	1882	17451

Le tableau ci-dessus montre les temps d'exécution absolus de LIMES et SILK. Tous les temps sont donnés en secondes. Les valeurs de la deuxième ligne du tableau sont les seuils de similarité θ . La figure ci-dessous montre les résultats de la comparaison des temps d'exécution relatifs de SILK et de LIMES. Les chiffres entre parenthèses dans la légende sont les valeurs du seuil θ :



LIMES est plus performant que SILK dans tous les paramètres expérimentaux. Il est important de noter que la différence de performance augmente avec la taille des bases de connaissances source et cible. Alors que LIMES ($\theta = 0,75$) nécessite environ 30% du temps de calcul de SILK pour l'expérience Drugs, il ne nécessite qu'environ 5% du temps de SILK pour les expériences SimCities. La différence de performance est encore plus significative lorsque le seuil est fixé plus haut. Par exemple, avec $\theta = 0,95$, LIMES ne nécessite que 1,6 % du temps d'exécution de SILK dans l'expérience SimCities.

3.2 Évaluation des fonctionnalités :

3.2.1 Format des entrées : Les deux outils offrent différents types de formats d'entrée pour les ensembles de données source et cible.

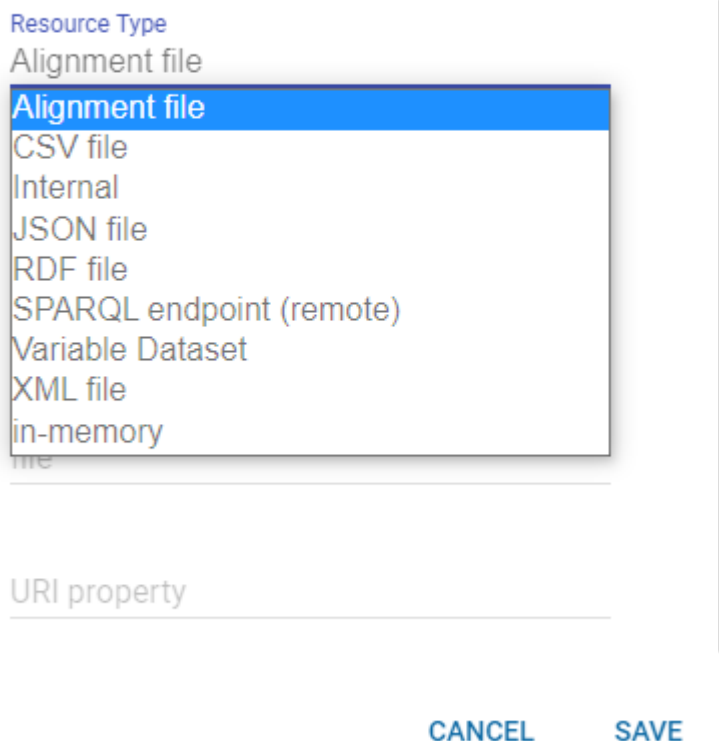
- LIMES** : LIMES accepte deux types d'entrées, le premier est un endpoint sparql et le second est un fichier local, dans le second cas la majorité des types sont proposés CSV, N3, TURTLE, etc.

Data source <input checked="" type="radio"/> Sparql endpoint <input type="radio"/> Local file Endpoint http://dbpedia.org/sparql Restriction ?s rdf:type urt:Movie Select restriction class Status of the request: OK	Data target <input type="radio"/> Sparql endpoint <input checked="" type="radio"/> Local file Endpoint Restriction ?t rdf:type dbpo:Film Select restriction class
---	---

- **SILK :**

Les principaux formats de données supportés par silk sont :

Create Dataset



Resource Type

Alignment file

Alignment file

CSV file

Internal

JSON file

RDF file

SPARQL endpoint (remote)

Variable Dataset

XML file

in-memory

URI property

CANCEL SAVE

il y a les fichiers csv, xml, rdf et json, il y a aussi les sparql endpoints (un point de présence sur un réseau HTTP qui est capable de recevoir et de traiter les demandes du protocole SPARQL et Il est identifié par une URL). En plus de cela, il y a les internals qui sont des datasets en mémoire qui aident à sauvegarder les entités entre les étapes de workflow. Le type InMemory qui est dataset qui contient toutes les données en mémoire. Variable dataset qui est un dataset qui sert de placeholder dans les flux de travail de Silk et qui est remplacé au moment de la demande. Alignment file qui permet d'écrire le format d'alignement spécifié à <http://alignapi.gforge.inria.fr/format.html> qui une API d'alignement est pour exprimer et partager des alignements d'ontologies.

3.2.2 Format des sorties :

- **LIMES :** Les formats de sortie proposés par limes sont :
 - TAB

- CSV
- NT
- TTL

TAB	
CSV	
TTL	able advanced options
N3	DISPLAY CONFIG

- **SILK** : Les formats de sortie proposés par limes sont :

Quand on crée une activité de transformation ou de liking, le résultat de ces opérations est généralement sauvegardé en mémoire ou bien dans l'un des dataset d'entrée.

Transformation Task

Source Dataset

Source Type

Source Restriction

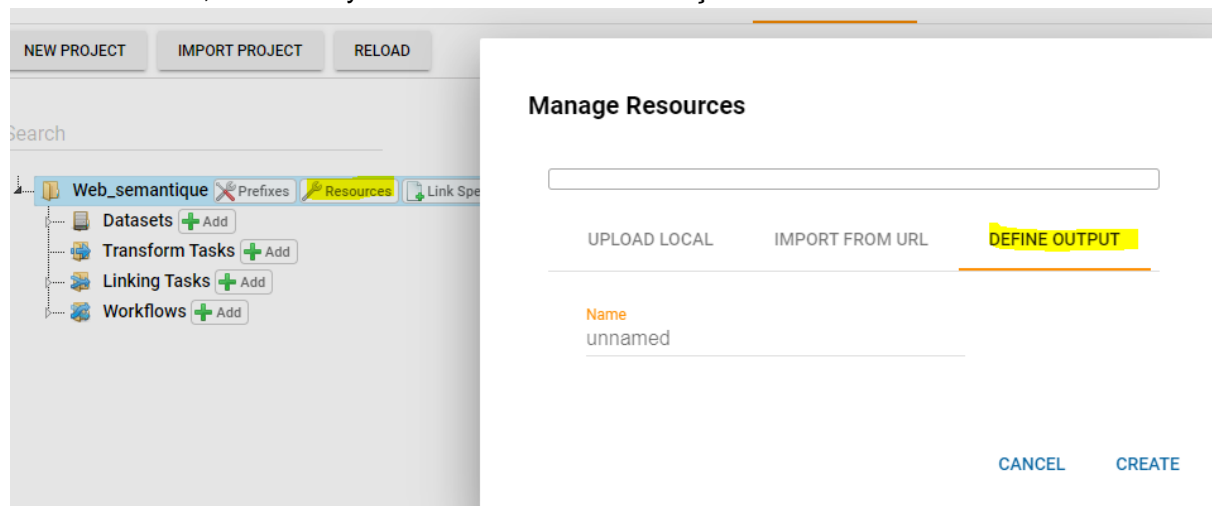
Output

london_crimes
londons_profiles

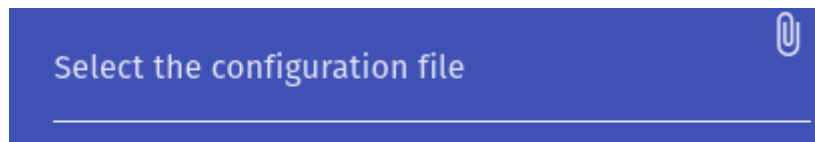
CANCEL

OK

Afin de sauvegarder ces résultats dans un fichier de sortie, il faudra le spécifier au niveau des ressources du projet et c'est à l'utilisateur de spécifier l'extension du fichier de sortie, dans notre cas, on a essayé avec l'extension .csv et ça a marché.



3.2.3 fichier de configuration : Les deux outils offrent des interfaces graphiques afin de spécifier les fonctionnalités nécessaires à l'utilisateur, les deux GUIs fonctionnent sans aucun bug et sont simples à utiliser, mais LIMES offre la possibilité d'utiliser un fichier de configuration, ce dernier est au format XML et LIMES l'utilise afin de savoir ce dont l'utilisateur a besoin.



3.2.4 Apprentissage automatique : Il n'est pas toujours facile de trouver la meilleure spécification de lien à utiliser pour découvrir les liens entre les ensembles de données source et cible. Pour cela limes offre un ensemble d'algorithmes d'apprentissage automatique à utiliser, ces derniers peuvent générer des spécifications de liens, il y a deux façons d'utiliser cette fonctionnalité : la première est d'utiliser l'interface utilisateur graphique et la seconde d'utiliser le fichier de configuration en remplaçant la balise METRIC par la balise MLAlgoritmh.

Machine Learning

Name

WOMBAT Simple

Type

unsupervised

rdfs:label

Source property

rdfs:label

Target property

max refinement tree size: 2000 × max iterations number: 3 × max iteration time in minutes: 20 × max execution time in minutes: 600 × max fitness threshold: 1 × minimum property coverage: 0.4 ×
 property learning rate: 0.9 × overall penalty weight: 0.5 × children penalty weight: 1 × complexity penalty weight: 1 × beta: 1 × verbose: false × atomic measures: jaccard, trigrams, cosine, qgrams ×
 save mapping: true ×

Parameter name

Value

max iteration time in minutes

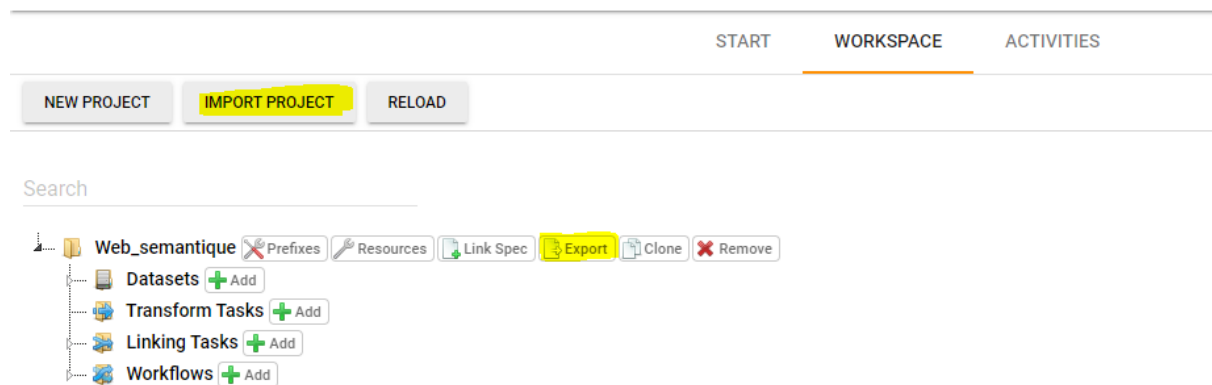
20

ADD

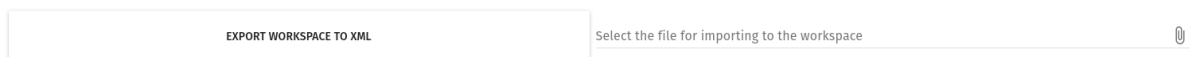
3.2.5 Exporter et importer un projet :

Les deux outils offrent les deux fonctionnalités aux utilisateurs Importer et Exporter.

Silk Workbench



Dans la figure ci-dessus, nous pouvons voir comment nous pouvons importer et exporter un projet dans SILK. LINES offre cette fonctionnalité, mais elle n'est pas aussi bonne que celle de SILK car dans LINES, nous ne pouvons exporter que les métriques.



3.2.6 Edition des préfixe :

Les deux outils offrent cette fonctionnalité, par exemple dans notre cas, on a ajouté le préfixe/vocabulaire cube qui permet de décrire notre graphe de données.

Edit Prefixes

Prefix

rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

rdfs: http://www.w3.org/2000/01/rdf-schema#

owl: http://www.w3.org/2002/07/owl#

qb: http://purl.org/linked-data/cube#

—

—

—

—

+

CANCEL

SAVE PREFIXES

Prefixes

owl url rdf dbpo

Label	Namespace
madsrdf	
bfic	
rdf	

ADD

3.2.7 Les fonctions de prétraitement : Les deux outils offrent un ensemble de fonctions pour le prétraitement.

- **LIMES** :

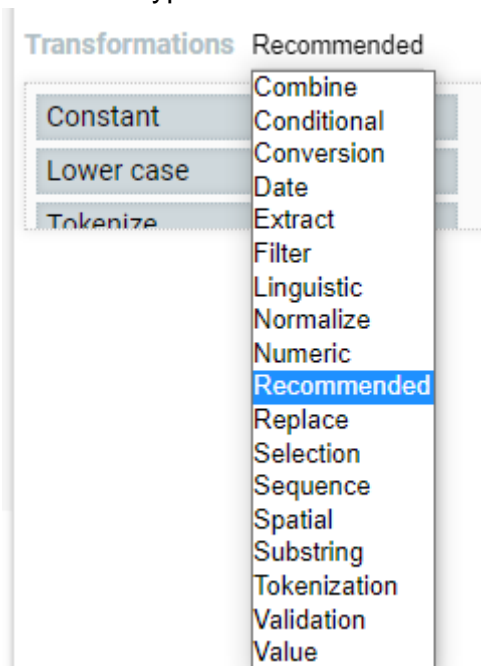
- **nolang** : pour supprimer les balises de langue.
- **lowercase** : pour convertir la chaîne d'entrée en minuscule.
- **uppercase** : pour convertir la chaîne d'entrée en majuscules.
- **number** : pour s'assurer que seuls les caractères numériques, "." et "," sont contenus dans la chaîne d'entrée.
- **replace(String a,String b)** : pour remplacer chaque occurrence de a par b.

- **regexreplace(String x,String b)** pour remplacer chaque occurrence de l'expression régulière x par b.
- **cleaniri** : pour supprimer tous les préfixes des IRIs.
- **celsius** : pour convertir Fahrenheit en Celsius.
- **fahrenheit** : pour convertir Celsius en Fahrenheit.
- **removebraces** : pour supprimer les accolades.
- **regularAlphabet** : pour supprimer les caractères non alphanumériques.
- **uriasstring** : renvoie la dernière partie d'un URI sous forme de chaîne.

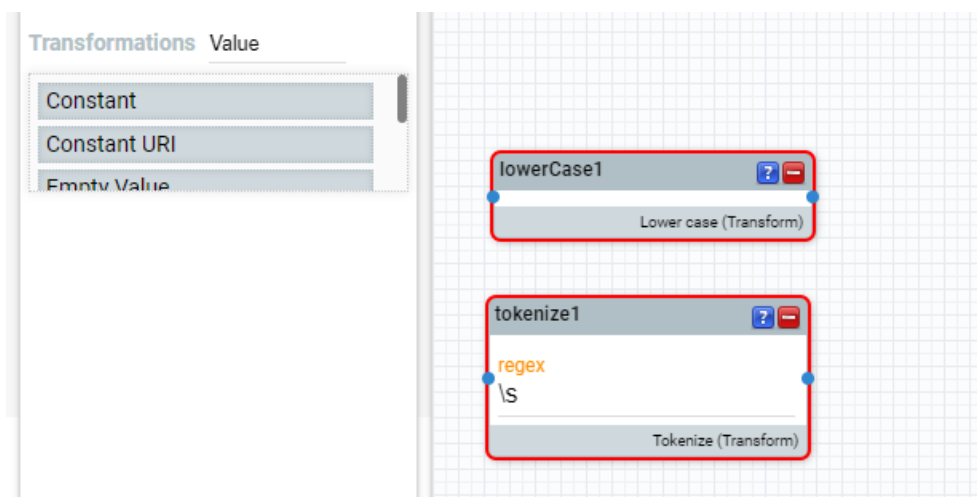
- **SILK :**

Voici les différentes fonctions qui permettent la transformation des données selon leur type :

on doit choisir tout d'abord le type des données



Après avoir choisi le type "recommandé", voici les fonctions de transformations les plus recommandées : lowercase et tokenize.



malgré les fonctions offertes par LIMES, mais SILK offre plus de fonctions par rapport à LIMES, nous pouvons voir que pour un seul type de données SILK offre plus que l'ensemble des fonctions de LIMES.

3.2.8 Les métriques de comparaison :

- **LIMES** : LIMES offre beaucoup de mesures, elles sont organisées en paquets, chaque paquet prend un type spécifique de données, les paquets sont :
 - **String measures** : Cosine, ExactMatch, Jaccard, Overlap, Jaro, JaroWinkler, Levenshtein, MongeElkan, RatcliffObershelp, Soundex, Koeln, DoubleMetaphone, Trigram et Qgrams.
 - **Vector space measures** : Euclidean et Manhattan.
 - **Point-set measures** : Geo_Hausdorff, Geo_Max, Geo_Min, Geo_Mean, Geo_Avg, Geo_Frechet, Geo_Sum_Of_Min, Geo_Naive_Surjection, Geo_Fair_Surjection et Geo_Link.
 - **Topological measures** : Top_Contains, Top_Covers, Top_Covered_By, Top_Crosses, Top_Disjoint, Top_Equals, Top_Intersects, Top_Overlaps, Top_Touches et Top_Within.
 - **Temporal measures** : Tmp_Concurrent, Tmp_Predecessor, Tmp_Successor, Tmp_After, Tmp_Before, Tmp_During, Tmp_During_Reverse, Tmp_Equals, Tmp_Finishes, Tmp_Is_Finished_By, Tmp_Overlaps, Tmp_Is_Overlapped_By, Tmp_Starts, Tmp_Is_Started_By, Tmp_Meets et Tmp_Is_xBy.
 - **Resource-set measures** : Set_Jaccard.
 - **Edge-counting semantic measures**

- **SILK** :

Il faut tout d'abord choisir le type de métrique à appliquer, comme dans ce cas là, on travaille avec les métrique basées sur les caractères :

The screenshot shows the SILK configuration interface. At the top is a 'Search term...' input field. Below it, 'Source Paths' is set to 'Crimes'. 'Target Paths' is set to a path containing a placeholder. A dropdown menu is open for 'Transformation', showing options: Asian, Characterbased (highlighted), Equality, Numeric, Recommended, Spatial, Temporal, and Tokenbased. Below the dropdown, 'Comparators' is set to 'Characterbased'. 'Aggregators' is set to 'Recommended'. At the bottom, 'qGrams' is selected for the transformation and 'Average' is selected for the aggregator.

Voici les métriques disponibles pour ce type de données (les caractères) :



Comme nous pouvons le voir, LIMEs offre beaucoup de fonctions de mesures par rapport à SILK, ce qui peut offrir beaucoup de flexibilité aux utilisateurs.

Tableau de synthèse de comparaison :

Fonctionnalité	SILK	LIMES
Input	Divers types de fichiers sont supportés par ces frameworks	
Output	Le choix des outputs est restreint par rapport à LIMES	LIMES offre plus de types de sorties par rapport à SILK qui stocke la sortie dans la mémoire.
Exporter et importer les projets	Cette fonctionnalité est disponible dans les deux frameworks.	
fichier de configuration	non supporté	LIMES offre deux façons de l'utiliser, la première est l'interface graphique et la seconde utilise le fichier de configuration.
Apprentissage automatique	non supporté	LIMES offre la possibilité d'utiliser des algorithmes d'apprentissage automatique dans le cas où il est difficile de définir la bonne spécification de lien.

Edition de préfixe	Elle est supportée par les deux frameworks	
Fonctions de transformation (pré-traitement)	Ces fonctions sont plus organisées au niveau de SILK (selon le type de données) et chaque type contient en moyenne 3 fonctions de transformation, ce qui fait que Silk a plus de fonctions de prétraitement.	
Métrique de comparaison	LIMES a plus de métriques de comparaison pour établir les liens entre les entités.	
Temps d'exécution	Silk offre aussi la fonctionnalité de Silk MapReduce qui est utilisée pour générer des liens RDF entre des ensembles de données en utilisant un cluster de plusieurs machines. Elle est basée sur Hadoop et peut par exemple être exécutée sur Amazon Elastic MapReduce. Silk MapReduce permet à Silk de s'adapter à de très datasets en distribuant la génération de liens à plusieurs machines.	D'après les trois expériences, LIMES offre une meilleure performance avec une grande différence par rapport à SILK. Mais il ne peut pas gérer un cluster afin de répartir les différentes tâches entre les serveurs.

Cas d'utilisation :

Nous allons maintenant utiliser les deux outils sur les mêmes jeux de données dans le but de comparer leurs performances et leurs résultats. La tâche consiste à trouver les mêmes entités qui apparaissent dans les deux jeux de données. Pour cela, nous allons utiliser le point de terminaison sparql de dbpedia pour accéder aux deux jeux de données. Ensuite, nous appliquons la transformation (lower_case) sur le label propriété, enfin nous utilisons la distance jaro pour comparer les deux labels transférés.

LIMES configuration :

- Nous avons d'abord ajouté les préfixes nécessaires : **owl**, **url**, **rdf**, **dbpo** et **rdfs**.
- Après cela, nous avons configuré les endpoints sparql, les deux étapes sont montrées dans la figure ci-dessous :

The screenshot shows the LIMES configuration interface. At the top, under 'Prefixes', there are buttons for 'owl', 'url', 'rdf', 'dbpo', and 'rdfs'. Below this is a table with columns 'Label', 'Namespace', and 'ADD'. The main configuration area is divided into two panels: 'Data source' and 'Data target'. Both panels have radio buttons for 'Sparql endpoint' (selected) and 'Local file'. The 'Data source' panel shows the endpoint 'http://dbpedia.org/sparql' and a restriction '7s rdf:type url:Movie'. The 'Data target' panel shows the endpoint 'http://dbpedia.org/sparql' and a restriction '7t rdf:type dbpo:Film'.

- Pour la métrique, nous utiliserons la métrique suivante :
`AND(cosine(s.rdfs:label,t.rdfs:label),exactmatch(s.rdfs:label,t.rdfs:label))`

Manual metric and machine learning

MANUAL METRIC MACHINE LEARNING

Source property PP
 Target property PP
 Optional source property PP
 Optional target property PP
 Cosine Threshold 0.5
 Source property
 Target property
 Operator AND
 cleaniri
 Rename As X
 Concat Rename A

Start Operator AND Cosine Threshold
 Source property Source property rdfs:label PP
 Target property Target property rdfs:label PP
 ExactMatch Threshold
 Source property Source property rdfs:label PP
 Target property Target property rdfs:label PP

Metrics

AND(cosine(s.rdfs:label,t.rdfs:label),exactmatch(s.rdfs:label,t.rdfs:label))

Maintenant Limes est prêt à commencer à travailler sur la tâche d'interconnexion. Les résultats de sortie seront enregistrés dans un fichier csv.

SILK configuration :

- Tout d'abord, nous avons créé un nouvel espace de travail, après cela, nous avons créé les ensembles de données source et cible en utilisant l'interface graphique.

Edit Dataset

Resource Type
 SPARQL endpoint (remote)

Dataset which retrieves all entities from a SPARQL endpoint

Name
 Films

endpoint URI
 http://dbpedia.org/sparql

login

password

graph

page size
 1000

entity list

CANCEL SAVE

- Puis, nous avons créé la tâche de liaison en utilisant également l'interface graphique.

Linking Task

Source Dataset
Movies

Source Type

Source Restriction
?a <http://www.w3.org/1999/02/22-rdf-synt

Target Dataset
Films

Target Type

Target Restriction
?a <http://www.w3.org/1999/02/22-rdf-synt

Output
Films

CANCEL

OK

- Maintenant SILK est prêt à commencer à travailler sur la tâche de liaison, la figure ci-dessous présente la configuration utilisée sur silk :

The screenshot displays the SILK configuration interface. It is organized into three main sections: Datasets, Transform Tasks, and Linking Tasks.

- Datasets:** Contains two datasets: **Films** and **Movies**.
 - Films:** Task identifier: Films; type: SPARQL endpoint (remote); pageSize: 1000; clearGraphBeforeExecution: true; pauseTime: 0; retryCount: 3; endpointURI: http://dbpedia.org/sparql; strategy: parallel; retryPause: 1000; useOrderBy: true.
 - Movies:** Task identifier: Movies; type: SPARQL endpoint (remote); pageSize: 1000; clearGraphBeforeExecution: true; pauseTime: 0; retryCount: 3; endpointURI: http://dbpedia.org/sparql; strategy: parallel; retryPause: 1000; useOrderBy: true.
- Transform Tasks:** Currently empty.
- Linking Tasks:** Contains one task: **Matching**.
 - Matching:** Task identifier: Matching; Source: Movies; Target: Films; Source Restriction: CustomOperator(?a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Movie> .); Target Restriction: CustomOperator(?a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Film> .).

Résultat :

Après avoir exécuté la tâche sur les deux outils, nous avons obtenu les résultats suivants :

LIMES :

- Pour le premier jeu de données, il a récupéré 141026 entités (292 seconds).
- Pour le second jeu de données, il a récupéré 165633 entités (308 seconds).
- Le mappage a été réalisé en 8.473 secondes avec 141382 entités liées.

SILK :

- Pour le premier jeu de données, il a récupéré 141026 entités (250 seconds).
- Pour le second jeu de données, il a récupéré 165633 entités (278 seconds).
- Le mappage a été réalisé en 16.938 secondes avec 150476 entités liées.

Une partie des résultats obtenus est présentée dans la figure ci-dessous :

<http://dbpedia.org/resource/The_Secret_of_Blood>	<http://dbpedia.org/resource/The_Secret_of_Blood>	1
<http://dbpedia.org/resource/The_Broadway_Melody>	<http://dbpedia.org/resource/The_Broadway_Melody>	1
<http://dbpedia.org/resource/Vaathiyaar_Veettu_Pillai>	<http://dbpedia.org/resource/Vaathiyaar_Veettu_Pillai>	1
<http://dbpedia.org/resource/A_Family_Affair_(1984_film)>	<http://dbpedia.org/resource/A_Family_Affair_(1984_film)>	1
<http://dbpedia.org/resource/Chetna>	<http://dbpedia.org/resource/Chetna>	1
<http://dbpedia.org/resource/Questi_pazzi_pazzi_italiani>	<http://dbpedia.org/resource/Questi_pazzi_pazzi_italiani>	1
<http://dbpedia.org/resource/Invisible_Agent>	<http://dbpedia.org/resource/Invisible_Agent>	1
<http://dbpedia.org/resource/Enemy_Territory_(film)>	<http://dbpedia.org/resource/Enemy_Territory_(film)>	1
<http://dbpedia.org/resource/The_Ladies_Man_(2000_film)>	<http://dbpedia.org/resource/The_Ladies_Man_(2000_film)>	1
<http://dbpedia.org/resource/Blood_Brothers_(film_series)>	<http://dbpedia.org/resource/Blood_Brothers_(film_series)>	1
<http://dbpedia.org/resource/Charlie_Chan's_Greatest_Case>	<http://dbpedia.org/resource/Charlie_Chan's_Greatest_Case>	1
<http://dbpedia.org/resource/At_the_Sound_of_the_Bugle>	<http://dbpedia.org/resource/At_the_Sound_of_the_Bugle>	1
<http://dbpedia.org/resource/Anumanaspadam>	<http://dbpedia.org/resource/Anumanaspadam>	1
<http://dbpedia.org/resource/Be_Yourself!>	<http://dbpedia.org/resource/Be_Yourself!>	1
<http://dbpedia.org/resource/I_Am_Angela>	<http://dbpedia.org/resource/I_Am_Angela>	1
<http://dbpedia.org/resource/I_Am_Fear>	<http://dbpedia.org/resource/I_Am_Fear>	1
<http://dbpedia.org/resource/Kindled_Courage>	<http://dbpedia.org/resource/Kindled_Courage>	1
<http://dbpedia.org/resource/The_Legend_of_Mor'du>	<http://dbpedia.org/resource/The_Legend_of_Mor'du>	1
<http://dbpedia.org/resource/Fiend_Without_a_Face>	<http://dbpedia.org/resource/Fiend_Without_a_Face>	1
<http://dbpedia.org/resource/Adventure_in_Morocco>	<http://dbpedia.org/resource/Adventure_in_Morocco>	1
<http://dbpedia.org/resource/Malice_(1926_film)>	<http://dbpedia.org/resource/Malice_(1926_film)>	1
<http://dbpedia.org/resource/The_Possible>	<http://dbpedia.org/resource/The_Possible>	1
<http://dbpedia.org/resource/Hoshi_Mamoru_Inu>	<http://dbpedia.org/resource/Hoshi_Mamoru_Inu>	1
<http://dbpedia.org/resource/Heidi_(2015_film)>	<http://dbpedia.org/resource/Heidi_(2015_film)>	1
<http://dbpedia.org/resource/Asterix_Versus_Caesar>	<http://dbpedia.org/resource/Asterix_Versus_Caesar>	1
<http://dbpedia.org/resource/Thodakkam>	<http://dbpedia.org/resource/Thodakkam>	1
<http://dbpedia.org/resource/Along_Unknown_Paths>	<http://dbpedia.org/resource/Along_Unknown_Paths>	1
<http://dbpedia.org/resource/Sons_of_Inngmar>	<http://dbpedia.org/resource/Sons_of_Inngmar>	1
<http://dbpedia.org/resource/The_Almighty_Dollar_(1923_film)>	<http://dbpedia.org/resource/The_Almighty_Dollar_(1923_film)>	1

Comme nous l'avons vu, LIMES a effectué la tâche en moins de temps que SILK avec une marge de 8s, mais SILK a détecté plus d'entités par rapport à LIMES, et il a été plus rapide dans la récupération des données.

Conclusion :

L'interconnexion des ensembles de données RDF est une opération importante pour les données liées (ouvertes) sur lesquelles la recherche est actuellement très. Du coup, ce projet nous a permis de découvrir les frameworks SILK et LIMES et voir comment ils arrivent à construire les liens d'interconnexion entre les datasets. À la fin, on a pu faire une synthèse de comparaison entre ces deux outils afin de voir les points faibles et les points forts de chacun.