SUMMARIZATION PIPELINE

1) ProcessOntology

2) splitting del dataset in 4 parti (1 per processore):
   1_lod_part_aa
   1_lod_part_ab
   1_lod_part_ac
   1_lod_part_ad
   e vengono salvati nella cartella ../datasetDIR/organized-splitted-deduplicated-tmp-file/

3) Ognuna delle 4 parti viene passato allo script awk. L'iesima istanza dello script (i in {1,..4}) genera i seguenti file:
   i*_types.nt
   i*_obj_properties.nt
   i*_dt_properties
   con * in Chars = {0,..,1}U{a,..,z}U{altri char}

   Avremo quindi 4|Chars| file.
   Questi file sono salvari in ../datasetDIR/organized-splitted-deduplicated

4) cancello i 4 file di splitting iniziail e la cartella ../datasetDIR/organized-splitted-deduplicated tmp-file/

5) i 4|Chars| file del punto 2 sono raggruppabili in gruppi di 4. Facciamo il raggruppamento in un unico file. Ad esempio:
   1d_dt_properties, 2d_dt_properties, 3d_dt_properties, 4d_dt_properties
   Vengono uniti in d_dt_properties e salvati nella cartella ../datasetDIR/organized-splitted deduplicated

6) Elimino i file del punto 3
   i*_types.nt
   i*_obj_properties.nt
   i*_dt_properties

   con i in {1,..4}
   con * in Chars = {0,..,1}U{a,..,z}U{altri char}

DA NOTARE CHE FINO A QUESTO PUNTO LE TRIPLE NON SONO ORDINATE MA SOLO DIVISE

7) Ordino parallelamente "sort -u" i file
   *_types.nt
   *_obj_properties.nt
   *_dt_properties

   con * in Chars = {0,..,1}U{a,..,z}U{altri char}

8) CalculateMinimalTypes

9) AggregateConceptCounts

10) ProcessDatatypeRelationAssertions

11) ProcessObjectRelationAssertions

12) CalculatePropertyMinimalization

13) Inference

14) CArdinalità

ProcessOntology: concetti e proprietà definiti dall'ontologia includendo anche le ontologie importate. Usa JENA.
- Scrivo su file l'elenco dei concetti dell'ontologia in reports/tmp-data-for-computation/Concepts.txt
- Scrivo su file l'elenco dei concetti dell'ontologia che sono sottoclassi di altri concetti in reports/tmp-data-for-computation/SubclassOf.txt

CalculateMinimalTypes: prede input i file datasetDir/organized-splitted-deduplicated/*_types.nt e li assegna a un pool di thread
Ogni thread ha un file da processare.
- Il thread i tiene una struttura con su tutti i concetti dell'ontologia. Intanto che legge le righe del file j aumenta il contatore ai concetti che vengono dichiarati come tipo di un'entità.
  Quindi la struttura mapperà ogni concetto dell'ontologia con le sue occorrenze nelle righe del file j.
  Scrive quindi il file /min-types/min-type-results/*_countConcepts.txt con * pari al prefisso * del file j. Ad esempio se è stato processato il file r_types.nt scriverà sul file r_countConcepts.txt.
- Scrive il file dei concetti esterni /min-types/min-type-results/*_uknHierConcept.txt
- Scrive i file dei minimal types di ogni entity del dataset: /min-types/min-type-results/*_minType.txt, naturalmente 1 file è scritto da un thread

AggregateConceptCounts:
- Nello step precedente abbiamo generato tanti file *_countConcepts.txt, ora dobbiamo aggregare il loro contenuti nel file patterns/count-concepts.txt. Viene quindi generato questo file che elenca tutti i concetti dell'ontologia e ci associa il numero di volte che il concetto j-esimo è stato dichiarato come tipo(ATTENZIONE, niente minimalità) di un'entità

ProcessObjectRelationAssertions: fase parallelizzata.
Lavora sull'insieme di file "/organized-splitted-deduplicated/*_dt_properties.nt"
Per ogni file:
  Preso l'insieme di triple nel file(che ha solo data type relational assertions) calcola
    - gli AKPs di ogni tripla e frequenze per ogni AKP. Alla fine scrive in patterns/count datatype.txt
    - le frequenze dei datatype degli oggetti di ogni tripla. Alla fine scrive in patterns/datatype akp.txt
    - le frequenze delle properties di ogni tripla. Alla fine scrive in patterns/count-datatype properties.txt

ProcessObjectRelationAssertions: fase parallelizzata.
Lavora sull'insieme di file /organized-splitted-deduplicated/*_obj_properties.nt"
Per ogni file:
Preso l'insieme di triple nel file(che ha solo objectrelational assertions) calcola
- gli AKPs di ogni tripla e frequenze oer ogni AKP. Alla fine scrive in patterns/object-akp.txt
- le frequenze delle properties di ogni tripla. Alla fine scrive in count-object-properties.txt

ARCHITETTURA ABSTAT

é modulare
Core:
- contiene algoritmo summarization.
- Gira da console.
- Torna summary in formato "proprietario" che poi andrà convertito in rdf affinchè sia indicizzato su virtuoso.

Web: interfaccia fruibile. Serve tutte le rotte, anche le API.

Virtuoso: contiene i summary e lo rende disponibile.

Solr: full-test search

NGINX: fa si che l'utente non sappia che è composto da diverse componenti. FA da punto d'accesso unico. Tutte le richeiste HTTP passano da nginx: questa è l'api la butto di là.. questa è una queery sparql la passo a virtuoso..
Fa l'instradamento. Se si vogliono esporre api in più si deve informare nginx

LIMITI E SOLUZIONI

- Sorting del preprocessing -> distribuzione
- navigazione summary usando virtuoso -> DB a grafo
- splitting -> sostituire .txt con DB