CalculateMinimalTypes
- legge ontologia
- estrae :
  concetti (concepts:Concepts),
  proprietà,
  relazione subClass (subclassRelations: List<String>),
  domain-range
  type graph (graph: TypeGraph)
- Pool di 10 thread, pesca file del tipo *_types.nt, finchè non finiscono ed esegue.
  Per ogni esecuzione:
  - genera una struttura (conceptCounts: HashMap<String, Integer>) a partire da concepts, perciò avrà TUTTI E SOLI I suoi concetti.
  - genera una struttura (externalConcepts: List<String>)
  - genera una strututra (minimalTypes: HashMap<String, HashSet<String>>)
  - per ogni riga del file, aggiorna il contatore di conceptCounts altrimenti aggiorna externalConcepts quindi aggiorna l'elenco associato all'entità corrente in minimalTypes eventualmente rimuovendo tipi non più minimi.
  - in fine scrive le occorrenze dei concetti di conceptCount in *_countConcepts.txt, externalConcepts in *_unHierConcept.txt e minimalTypes in *_minType.txt

AggregateConceptCounts
- crea un counts:ConceptCount il quale contiene una struttura che mappa concetti con occorrenze (conceptCounts: HashMap<String, Long>)
- in modo sequenziale, per ogni file *_countConcepts.txt aggiorna conceptCounts: ogni volta che legge un concetto e il suo count, aggiunge tale valore a quello già presente per quel concetto in ConceptCounts
- scrive ConceptCount nel file patterns/count-concepts.txt

ProcessDatatypeRelations
- Crea un oggetto counts:OverallDatatypeRelationsCounting che contiene 3 oggetti di tipo Vector<NTripleAnalysis>: datatypesCount, propertiesCount, akpCounts
- Pool di 10 thread, pesca file del tipo *_dt_properties.nt, finchè non finiscono ed esegue. Per ogni esecuzione:
  - crea un oggetto analysis:DatatypeCount che contiene una mappa (counts:HashMap<String, Long>)
  - crea un oggetto propertyCount:PropertyCount che contiene una mappa (counts:HashMap<String, Long>)
  - crea un oggetto akpCount:AKPDatatypeCount che riceve il file con I tipi minimi che nel suo nome ha lo stesso prefisso del file corrente. Quindi mette I tipi minimi in una mappa (types:HashMap<String, List<String>>).AkpCount ha anche una mappa akps:HashMap<String, Long>
  - per ogni riga del file:
    la si converte in un oggetto Ntriple e viene analizzata:
    . da analysis, per contare le occorrenze dei datatype aggiornando analysis.counts
    . da propertyCount, per contare le occorrenze dei predicati aggiornando propertyCount.counts
    . da akpCount, il quale cicla per ogni tipo minimo del subject cercandoli in akpCount.types (se no ha tipo minimo usa Thing), costruisce l'akp e lo tiene in akpCount.akps:HashMap<String, Long> aggiornandone il contatore.
  - aggiungo analysis a datatypesCount, aggiungo propertyCount a propertiesCount e agigungo akpCount a akpCounts
- finito di analizzare ogni file
  - aggrega gli HashMap degli elementi di datatypesCount e scrive in patterns/count-datatype.txt
  - aggrega gli HashMap degli elementi di propertiesCount e scrive in patterns/count-datatype-properties.txt
  - aggrega gli HashMap degli elementi di akpCounts e scrive in patterns/datatype-akp.txt

ProcessObjectRelations

- Crea un oggetto counts:OverallObjectRelationsCounting che contiene 2 oggetti di tipo Vector<NTripleAnalysis>: propertiesCount, akpCounts. Contiene anche un oggetto minimalTypesOracle:AllMinimalTypes. MinimalTypesOracle ha una struttura types:HashMap<String, PartitionedMinimalTypes> che contiene le info di TUTTI I file _minTypes.txt: la chiave è il prefisso * mentre il valore è un oggetto che ha una mappa<entità, tipi minimi>. MinimalTypes ha anche others:<entità, tipi minimi>.

- Pool di 10 thread, pesca file del tipo *_obj_properties.nt, finchè non finiscono ed esegue. Per ogni esecuzione:
  - crea un oggetto propertyCount:PropertyCount che contiene una mappa (counts:HashMap<String, Long>)
  - crea un oggetto akpCount:AKPObjectCount che riceve MinimalTypesOracle .AkpObjectCount ha anche una mappa akps:HashMap<String, Long>
  - per ogni riga del file:
    la si converte in un oggetto Ntriple e viene analizzata:
    . da propertyCount, per contare le occorrenze dei predicati aggiornando propertyCount.counts
    . da akpCount, il quale cicla per ogni tipo minimo del subject cercandoli in akpCount.types (se no ha tipo minimo usa Thing), cicla poi internamente sui tipi minimali dell'object e costruisce gli akp (come prodotto cartesiano), aggiorna quindi I contatori di akpCount.akps:HashMap<String, Long> e infine scrive su object-akp_grezzo.txt la riga <tripla>[AKPs]
  - aggiungo propertyCount a propertiesCount e aggiungo akpCount a akpCounts
- finito di analizzare ogni file:
  - aggrega gli HashMap degli elementi di propertiesCount e scrive in patterns/count-object-properties.txt
  - aggrega gli HashMap degli elementi di akpCounts e scrive in patterns/object+-akp.txt