A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

Phase 1: alle Klassen warden gezählt, für jede entity werden die parent klassen aufgeschrieben

Auch in Parser.entityExtraction (aufgerufen von SelectionView)

Warum wird in phase 2 nicht auf die nicht typepredicate gefiltert? -> ist dann halt auch drinnen mit isType beziehung

Phase 2: objTypesIDs sind nur die distinct Zielklassen

Befüllt entityDataHashMap: für jede entity sind die überklassen gespeichert. In der propertyconstraintmap sind jeweils die beziehungen gespeichert und dann alle überklassen, auf die die beziehung geht. Der counter gibt die menge an der beziehung an.

In classtoPropWithObjTypes gibt es jede Klasse einmal. Dann gibt es eine liste an relationships mit einer liste an zielen (überklassen).

Phase 3: geht entityhashmap durch und rechnet in einer map supp und conf aus. Für jeden typ, jeden relationship-typ & object

Schritt 3 und schritt 4 noch nicht angesehen. Auch approx noch nicht durchgegangen.

File name notiation: 0.2 confidence (20%)\_200 support

* Opt. 1 -> von thema 1 dazupfuschen
  + eher nicht
* Opt. 2 -> vergleich von graphen diff
  + warum nicht, laufzeiten vergleichen
  + Würde einen Graph und ein Diff bekommen und die Shapes von Graph1.
  + Todo research wie ein graph diff aussieht -> jena
  + Die kann man dann in den alg einfließen lassen. In alle? Oder nur in exact oder in alle varianten?
  + Laufzeiten vergleichen geht noch immer
  + Auch kombinierbar mit website -> man kann neue version oder diff hochladen
  + Laufzeit bei diff berechnung? Muss ja dann in graph umgewandelt werden
* Opt. 3 -> recherchieren warum shape hinausfällt
  + nicht schwer
  + entweder, weil sich parameter verändert haben (einfach infos mitnehmen von support und confidence) oder die daten anders sind
  + shapes können nur dazukommen oder wegfallen
  + auch verändern? -> ja, z.b. min max cardinalities
* Opt. 4 -> wie schon vorhandene shapes in den algorithmus mithineingenommen werden können
  + Wäre auch eine option, dauert sicher ein bisschen
  + Man könnte ein paar Schritte im Algorithmus überspringen
  + Aber mit welchen parametern? Sind die dann fix anzunehmen, weil eigentlich müsste man es ja nochmal überprüfen
  + Mit support und conf von algorithmus davor, wäre es eigentlich machbar -> die kann man eh inkludieren von algorithmus aus
  + Mit unterschiedlichen parametern oder denselben?
    - Mit denselben bleibt eigentlich nichts anderes als es nochmal zu laufen
    - Mit unterschiedlichen könnte man, die die sicher noch drinnen sind, drinnen lassen
    - Eigentlich kann es nur in schritt 4 angewendet werden,
* Opt. 5 -> nur Website -> ein bisschen zu wenig
  + Website bauen, man kann graph anlegen, dann mehrere versionen hochladen.
  + Für jede version können shapes generiert werden (mit qse) -> die shapes dann vergleichen. Man kann parameter für jeden lauf eingeben.
  + Vergleich zwischen verschiedenen parametern und graphversionen?
  + Prozent gleich
  + Text unterschiede
  + Todo shactor ansehen wie evaluiert -> gar nicht, muss ich selbst in rq machen

Mit welchen datasets

Interviews danach?

Welche algorithmen?

Wie bekomme ich veränderte datensets?

Wie sehr verändert sich graph? Kann auch sein, dass er komplett anders ist

Todo tag ansehen von git und sehen, ob da nur eine version drinnen ist -> jar file, docker usw -> nein Nicht viel unterschied zwischen flag vldb qse und qse jetzt

Todo Docker usw ausführen

Todo mail schreiben -> dann exposee schreiben, wie am besten graphs vergleichen

Warum gibt es keine unit tests?

Welche methode wird für approx verwendet und zu was braucht man die anderen klassen?

Warum kann man von der main approx nicht ausführen? Warum werden die parameter nicht verwendet?

Warum Code in Java geschrieben? -> alternative python, hat beides vor- und nachteile

Analyse nirgends verwendet (außer bei Parameter qse\_validation)

StandardReservoirSampling nirgends verwendet

qse\_approximate\_file -> ReservoirSamplingParser

Neighbourbased nirgends verwendet

DynamicNeighborBasedReservoirSampling wird verwendet

Wahrscheinlich einfach vergessen, Main zu ändern

Andere überlegungen: alles in pyhton umschreiben (kann ich nicht), tests schreiben (sollte nicht meine aufgabe sein)

Next steps: einmal mit docker ausführen (mit unverändertem Main)

Kg videos ansehen, papers lesen, literature review, themen fragen klären (details), rq definieren

Hat es Sinn, sich mit Evolving KGs zu beschäftigen? Die sind noch relativ neu, kann noch viel kommen.

Nur für graphdb? Da gibt es hist plugin schon, das wäre ziemlich praktisch. Aber wie komme ich da an daten?

Changesest xml?

Rdf star ignorieren?

Named graphs -> snapshot

Versioning vocabulary -> VoId oder ProvO

Reification

Timestamps and provenance

Welche daten bekomme ich? Kommt ja auch darauf an. Darauf muss ich die logik umändern, das ist ja pro art unterschiedlich.

Shacl temporal constraints

Difference shacl and owl

Ganz andere ansätze in papers z.b. molecules, die zusammenfassung bieten

Möglichkeiten versioned graphs:

* Named graphs (snapshot)
  + Rdf source
* Versioning vocabulary void, provo
  + Provo ist eine ontology, aber hat keywords wie generatedby, revisionof, previous\_version; wahrscheinlich ziemlich kompliziert….
* Timestamps and provenance
  + Ähnlich wie mit anderen attributen; nur dass timestamp angegeben wird
* Reification
  + Wie in vorlesung, ähnlich wie bei versioning mit void und provo
* Graphql
  + Wäre ziemlich praktisch
  + Nicht für große datensätze
* Changesetxml -> veraltet eher RDF Patch oder sparql update; -> eher eine art, um festzustellen, was geändert worden ist
* Rdf star ignorieren

Todo jede art recherchieren, wie diff recherchieren, welche art standard ist; papers lesen zu evolution