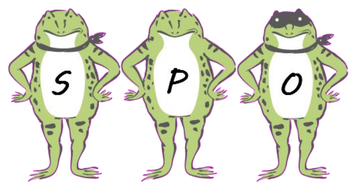
大規模RDFグラフに対するデータ圧縮と検索高速化の両立

兼岩研究室　藤原　浩司

RDFトリプルとトリプルパターン

【RDFトリプルとは？】

主語(subject), 述語(predicate), 目的語(object) の3つ組の情報表現。
例：電気通信大学についてのRDFトリプル



- 電気通信大学, rdf:type, 国立大学
- 情報理工学部, 学部, 電気通信大学
- 電気通信学部, 学部, 電気通信大学

RDFトリプルの集合はグラフ構造(RDFグラフ)を構成する

【トリプルパターンとは？】

特定のRDFトリプルを得るために使うパターン。
RDFトリプルに変数を加えたもの。
例：上のRDFグラフから、電気通信大学の学部を得たいとき

- ?x, 学部, 電気通信大学

このトリプルパターンを使うと変数?xに対して

- ?x = {情報理工学部, 電気通信学部}

という結果が得られる。

【Web上のRDFデータ】

Web上のRDFデータの規模は膨大で、どんどん増大している

- DBPedia Japanese　：約8000万トリプル
- UniProt　　：約300億トリプル

目的

大量のRDFデータを高速に処理！

提案手法

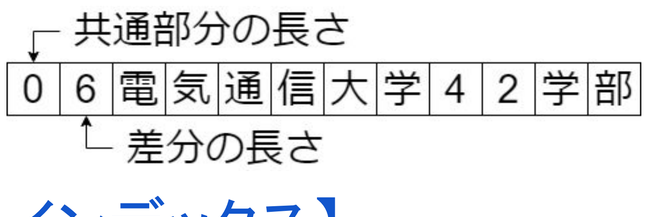
【辞書の作成と圧縮】

トリプルに出現する文字列をソートしてIDを振り、IDと文字列の対応表(辞書)を作る。トリプルはIDで管理する。
例：上記の電気通信大学のトリプルの場合(概略図)

- 5, 1, 2
- 4, 3, 5
- 6, 3, 5

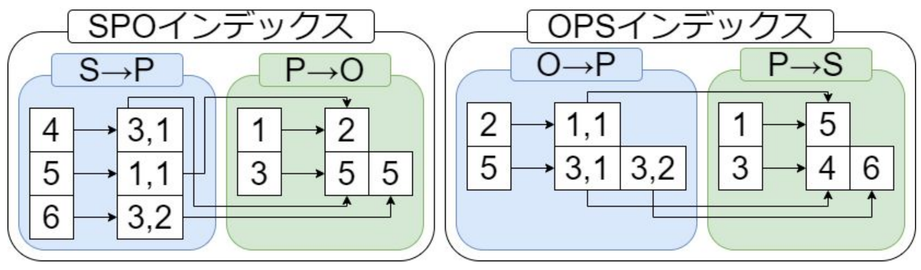
1	rdf:type	4	情報理工学部
2	国立大学	5	電気通信大学
3	学部	6	電気通信学部

また、辞書を文字列同士の差分で圧縮する手法(FrontCoding)を用いて圧縮する。
例：「電気通信大学」と「電気通信学部」の差分圧縮



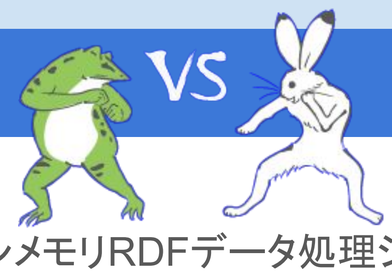
【リレーインデックス】

省スペースかつトリプルパターンの高速検索が可能なデータ構造『リレーインデックス』を提案。
例：上記の電気通信大学のトリプルの場合(概略図)



SPOインデックスとOPSインデックスという2つの構造だけですべてのトリプルパターンの検索を効率的に処理可能。

評価



【実行環境】

提案手法を実装したインメモリRDFデータ処理システム、『FROST』をJavaで実装。
データ圧縮と検索処理の評価実験を以下のマシンで実施

- OS：Windws 8.1 Enterprise 64bit
- CPU：Intel(R) Xeon(R) CPU E5-2667 v2@3.30GHz 3.30GHz
- メモリ：128.0GB

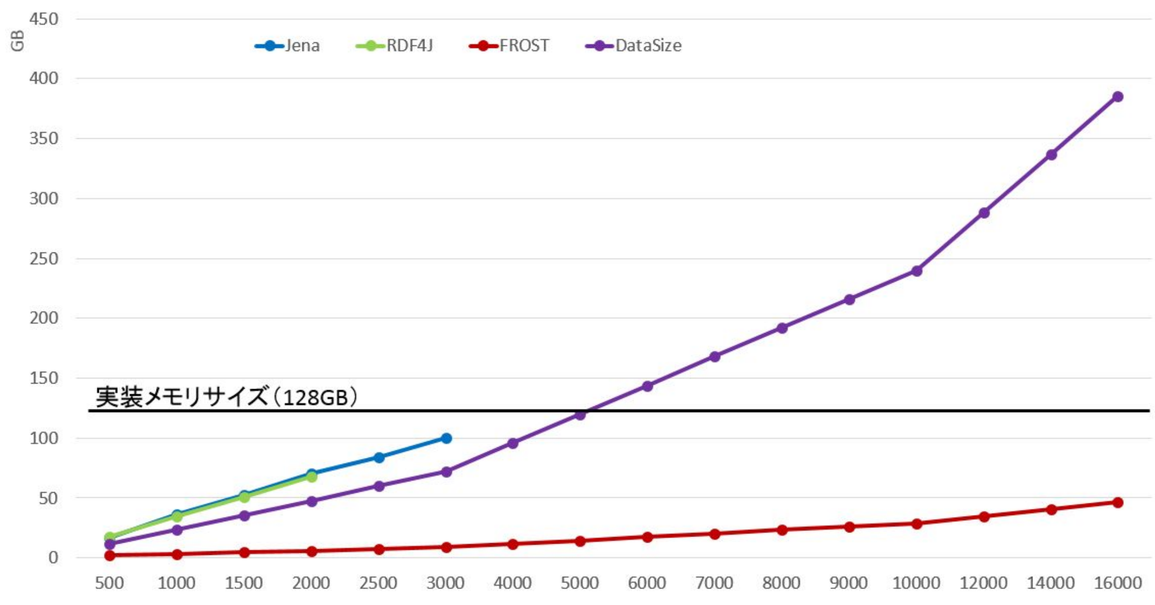
比較対象

- インメモリ：Jena 3.1.1、RDF4J 2.1.4
- オンディスク：Virtuoso 7.2.4

データはベンチマーク用のデータが生成可能なLUBMを使用

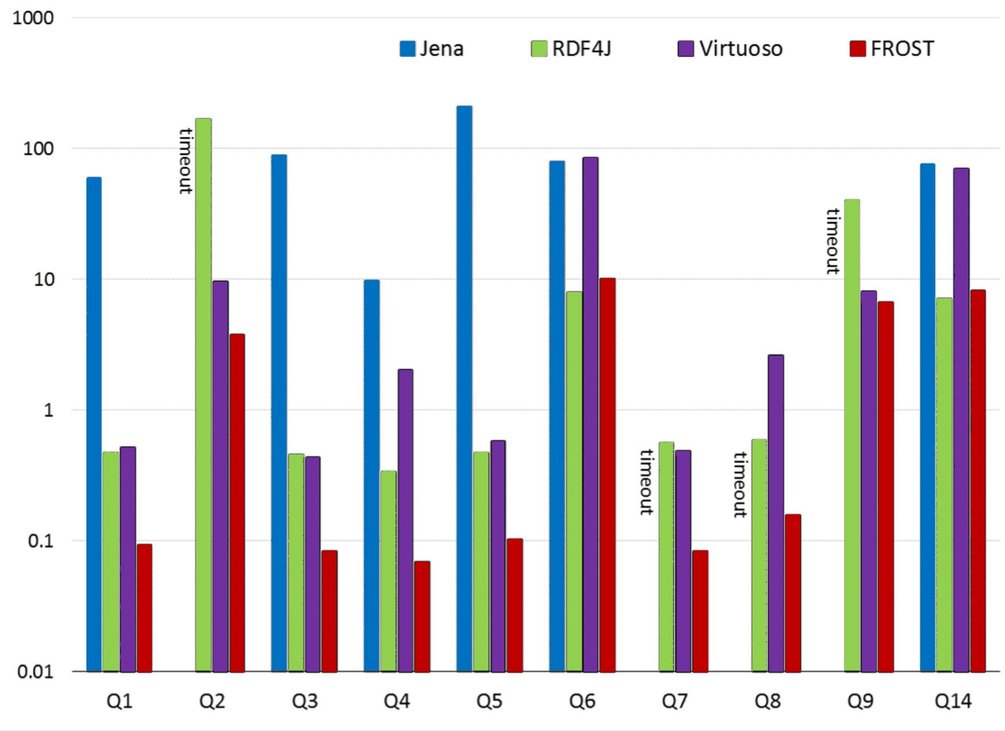
【データ圧縮の評価】

LUBMのサイズを大きくした時のメモリ使用量を比較。
実装メモリサイズよりもはるかに大きなデータを読込可能。



【検索処理の評価】

LUBMのベンチマーククエリの実行時間を比較。
ほとんどのクエリで他システムよりも短時間に処理可能。



今後の展望

【圧縮/検索性能の向上】

- 1つのインデックス構造だけで各パターンを処理する
- 同型グラフやルール化による圧縮

【機能の追加】

- 名前付きグラフへの対応
- Datalog推論によるRDFグラフの拡張
- 並列処理によるデータ処理
- IoTデバイスへの搭載

データ構造やアルゴリズムに興味のある方、お待ちしています！