Ranking RDF properties using Spark framework

Akmal Khikmatullaev, Seyithan Dag

Winter Semester 27/02/2018

Outline

- Problem statement
- Paper background
- Implementation
  - Adopted work
  - Data preparation
  - Algorithms used
- Results
- Conclusion and Future Work

Problem statement

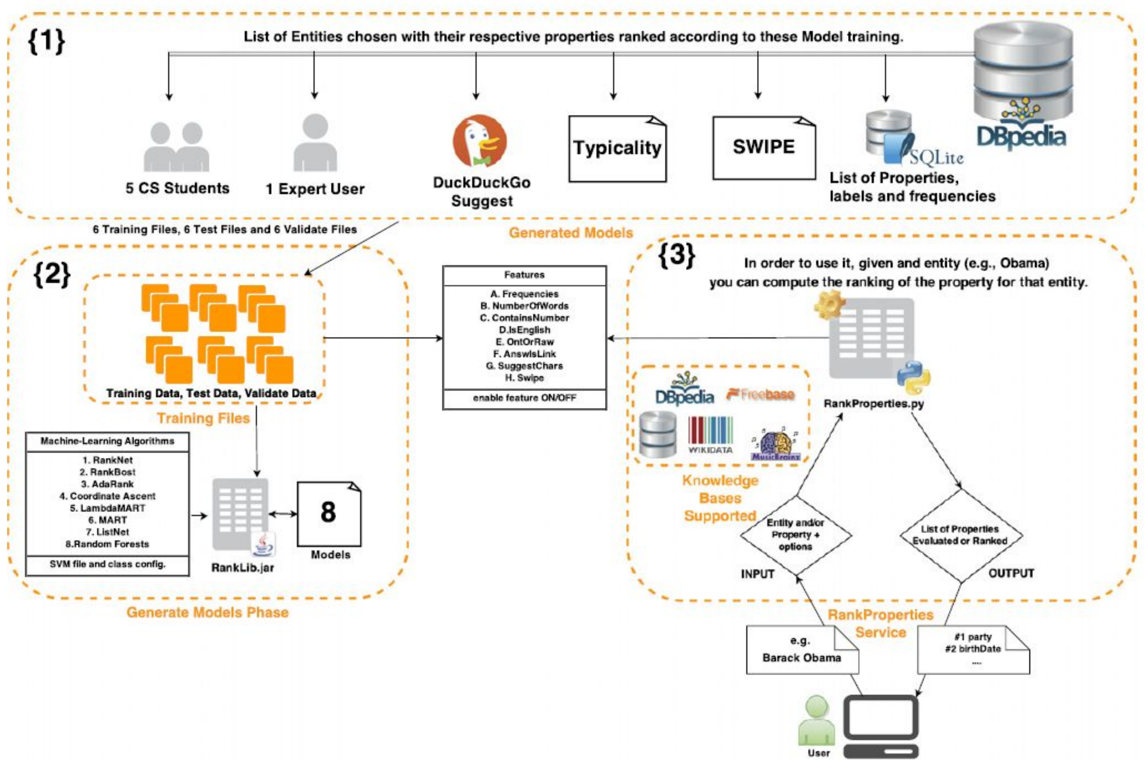
Problem statement

Problem: providing an order of relevance, or in short, ranking RDF properties/predicates.

<subject> <predicate> <object>

Paper background

Paper background



Paper background: Features

A. ***Frequency*** - Frequency is generally the most used feature for ranking.
B. ***NumberOfWords***: we defined to help the ranking, is the count of the words contained in theproperty's name.
C. ***ContainsNumber***: Similar to the previous feature,also the presence of numbers in the URI can give insights about the quality of the property.
D. ***IsEnglish***: Following the same argument used in the Contains Number feature, property names(or labels) can be more or less meaningful.
E. ***OntOrRaw***: This DBpedia specific binary feature determines whether the feature is in the DBpedia ontology (prefix dbpedia-owl) or not.
F. ***AnswIsLink***: Another discriminant is given by the range of the property.
G. ***SuggestChars***: This value measures how popular is the name of the property w.r.t. the provided entity, according to DuckDuckGo suggestion.
H. ***Swipe***: implemented an API that provides a list of the most important properties used in SWiPE, set 1 if the property in the infobox otherwise 0.

Paper background: Training set preparation

The authors use 6 ways of preparing the training data:

- ❑ Expert-based
- ❑ Questionnaire-based
- ❑ Frequency based
- ❑ Suggest training, using DuckDuckGo
- ❑ Typicality based
- ❑ SWIPE based

Paper background: ML Algorithms

The library used by the authors is RankLib. It includes:

- ❑ ***RankNet***: a neural network based on a simple probabilistic cost function to model underlying ranking
- ❑ ***RankBoost***: an algorithm that is trained on pairs, trying to solve the preference problem directly
- ❑ ***AdaRank***: a method for direct optimization of performance measures
- ❑ ***Coordinate ascent***: algorithm for multivariate objective function optimization
- ❑ ***LambdaMart***: learning to rank algorithm, based on Multiple Additive Regression Tree
- ❑ ***MART***: gradient tree boosting method implementation, for predictive data mining
- ❑ ***ListNet***: neural network with gradient descent optimization, optimizing list-wise loss based on probability
- ❑ ***Random forests***: collection of decision trees

Implementation

Implementation: Adopted work

- ***3 domains were decided on***: language, country and city.
- ***Training data generation method***: frequency-based training.
- ***Features used***: frequency, number of words, contains number, is English, ontology or raw, object (answer) is link.

  Suggested chars and SWIPE features were not used.
- ***Algorithms used***: multinomial logistic regression, naive bayes, decision trees, and isotonic regression.

Implementation: Data preparation

- ***Language:***
  - 921,715 RDF triples
  - 1,242 distinct predicates
- ***Country***:
  - 1,178,959 RDF triples
  - 2,545 distinct properties
- ***City***:
  - 5,000,000 RDF triples
  - 3,852 distinct properties

Implementation: Data preparation

Example SPARQL query for retrieving all RDF triples about “City”.

```
prefix dbpedia-owl: <http://dbpedia.org/ontology/>

select ?entry ?p ?o {{
  select ?entry ?p ?o{{
    ?entry a dbpedia-owl:City.
    ?entry ?p ?o
  }
  order by ?entry
}}

limit
offset
```

Implementation: Data preparation

Frequency based training data generation: 4 ranks were used as labels, where the 1st rank is the most valuable. For 100 predicates, for the ith predicate with frequency $f(i)$:

$0 <= f(i) < 25$: rank 4

$25 <= f(i) < 50$: rank 3

$50 <= f(i) < 75$: rank 2

$75 <= f(i)$: rank 1

Implementation: Algorithms

- ❑ Multinomial logistic regression
- ❑ Naive bayes
- ❑ Decision tree
- ❑ Isotonic regression

All of the data was turned into RDDs, then the features were extracted. After that, each algorithm was trained using all data belonging to all of the selected domains.

Results

Results

In each of trainings, 60% of the data was used for training, and 40% was used for testing. Following accuracies were obtained:

algorithm / sphere	multinomial logistic reg.	naive bayes	decision tree	isotonic regression
language	0,30	0,66	0,83	~0,01
country	0,36	0,65	0,85	~0,01
city	0,36	0,70	0,84	~0,01

Conclusion and Future Work

Conclusions and Future work

- Different strategies to ranking RDF properties, based on a MLR framework was explored, one of them was implemented.
- Parallelized the training process, even with such numbers of predicates, the whole process executed in a timely fashion.
- As expected, isotonic regression proved useless, as they are incapable for multilabel classification. If the rankings are turned into probabilities, they might work. Decision trees proved most useful.
- As a futurework, exploration of positive outcomes of appropriate RDF ranking in different applications such as Question Answering can be done.
- RDF property ranking can be included in the semantic web as a dynamic feature.

Thank you for your attention!