# Heuristic based ranking of properties.

Open Universiteit   
Faculteit Informatica

14-5-2015

versie 1.0

J. N. Theunissen

838573218

Inhoud

[Heuristic based ranking of properties. 1](#_Toc418456984)

[Inleiding. 3](#_Toc418456985)

[Infobox heuristic sort methode 3](#_Toc418456986)

[In bucket ordening 4](#_Toc418456987)

[Term frequentie ordening 5](#_Toc418456988)

[Range rank calculation 5](#_Toc418456989)

[Opzet inplementatie van het algorithme 6](#_Toc418456990)

[Inplementatie in Java 6](#_Toc418456991)

[Alfabetische sortering implementeren. 6](#_Toc418456992)

[Heuristic sortering implementeren. 7](#_Toc418456993)

[Klasse opzet 7](#_Toc418456994)

[Informatie benodigdheden voor de heuristic 7](#_Toc418456995)

[Range rank berekening 8](#_Toc418456996)

[Inventarisatie van de gegevens. 9](#_Toc418456997)

[Implementatiewijzigingen 9](#_Toc418456998)

[References 11](#_Toc418456999)

# Inleiding.

Dit document beschrijft hoe de heuristic sort methode zoals het wordt gepresenteerd in de   
master scriptie van Paul Falco (Paul, 2014) geïmplementeerd is in de Fesnel Forms plug-in.

De master scriptie van Paul Falco evalueert van verschillende sorteerd algoritmes welke het beste lijkt op de sorteer volgorde van de infoboxes van wikipedia. De heuristic methode komt het dichtst hierbij in de buurt.

# Infobox heuristic sort methode

Deze heuristic methode bestaat uit een aantal algemeen geldende vuistregels. Deze regels worden hieronder beschreven:

In tabel 1 staan de regels en de volgorde waarin de regels uitgevoerd moeten worden.

Tabel : De sorteer regels.

|  |  |  |
| --- | --- | --- |
| Volgorde | Bucket | Omschrijving |
| 1 | A | Any property which contains one of the following strings in its label is promoted  to bucket *A*:   ‘name’   ‘code’   ‘identifying’   ‘identifier’   ‘label’ |
| 2 | E | Any properties from the following ontologies are promoted to bucket *E*:   foaf:name   foaf:nick   foaf:givenName   foaf:givenname   foaf:familyName   foaf:family\_name   foaf:firstName   foaf:lastName   foaf:surname   foaf:topic   http://schema.org/alternateName   http://schema.org/name   dc:title   dc:subject   dc:identifier |
| 3 | F | Properties from the following ontologies will be promoted to bucket *F*:   foaf:homepage   foaf:isPrimaryTopicOf   foaf:primaryTopic   foaf:knows   foaf:made   foaf:maker   foaf:page   foaf:gender   http://schema.org/description   http://schema.org/sameas   http://schema.org/url   http://schema.org/about   http://schema.org/category   http://schema.org/object   dc:creator   dc:description   dc:type   dct:alternative |
| 4 | B | If the property is any of the following XSD date data types then bucket B is  assigned:   XSDdate   XSDdateTime   XSDtime   XSDgYearMonth   XSDgYear   XSDgMonthDay   XSDgMonth   XSDgDay |
| 5 | C | We then look at the range of the property and compute a metric that we call  the range rank. If a range rank has been assigned than the property is promoted to bucket *C*. The actual range rank becomes the immediate next level of ordering within  bucket *C* (popular ranges are ranked higher).  See below for a calculation of the range rank. |
| 6 | D | Properties that are either functional or inverse functional are assigned to  bucket *D*. |
| 7 | Z | Any other properties are assigned to bucket *Z* (these will be the lowest ranking  properties). |

## In bucket ordening

De in bucket ordering vindt plaats op basis van de property label.

Tabel : De in-bucket sorteer regels.

|  |  |
| --- | --- |
| Volgorde | Omschrijving |
| a | The highest priority (*a*) is assigned to properties that have labels containing  any of the following words: “start”, “begin”, “birth”, “former”, “previous”,  “opening” or “predecessor”. |
| b | The second priority (*b*) is set for those properties where the label contains either  “stop”, “end”, death”, “future”, “subsequent”, “closing” or “successor”. |
| c | The lowest priority (*c*) is designated to all properties that do not fall in one of  the first two categories. |

## Term frequentie ordening

Binnen de bucket worden gelijke geordende termen nog geordend op frequentie.  
Frequenter gebruikte termen komen eerst. Bij gelijke frequentie wordt een alfabetische volgorde toegepast.

## Range rank calculation

Nadat een property in een bucket op de juiste volgorde is geplaatst wordt de rank berekend.  
Voor dataproperties volgt een ranking van 0 voor de onderstaande XSD datatypes.

 XSDbyte

 XSDdouble

 XSDfloat

 XSDdecimal

 XSDboolean

 XSDint

 XSDshort

 XSDstring

 XSDunsignedByte

 XSDunsignedInt

 XSDunsignedLong

 XSDunsignedShort

 XSDnonNegativeInteger

 XSDnegativeInteger

 XSDnonPositiveInteger

 XSDpositiveInteger

 XSDduration

Voor object ranges wordt wordt de rank in 2 stappen berekend.

1. Er wordt een geordende lijst aangemaakt voor alle klasse in de ontologie.  
   Voor elke van de klasse wordt bepaald hoevaak ze in een range gebruikt worden. De ordening in de lijst loopt van klasse die het meest naar de klasse die het minst gebruikt worden in een range.
2. De range rank van een property wordt berekend door van de range object een lijst te maken van de klasse in de overervings hiërarchie. De range rank is nu de klasse uit de overervings hiërarchie met de hoogste populariteit. De index uit de rangeclasses list is de rangerank. Op deze manier zou een lijst samengesteld kunnen worden van de properties met de range rank.

# Opzet implementatie van het algoritme

Het rechtermuismenu van de lens krijgt twee sorteer opties: alfabetisch sorteren en heuristic sorteren. Dit om het verschil te zien tussen beide methodes.

Voor de sorteer actie wordt een nieuwe comparator opgezet die twee property objecten kan vergelijken. De twee property objecten worden voorzien van de bucket waarin ze vallen.  
Zijn de buckets ongelijk, dan bepaalt het ordinal getal van de buckets de uitkomst van de comparator. Zijn de buckets gelijk dan wordt de ordening binnen de buckets bepaalt en die bepaald dan de uitkomst van de comparator.

# Implementatie in Java

De properties van een lens worden weergegeven door de LensBox klasse.  
Elke lensbox heeft een arrayList ‘propertylabels’.

De sortering wordt uitgevoerd door de statische methode Collections.sort().  
Deze methode verwacht een vergelijker (Comparator). Deze vergelijker bepaalt de sorteervolgorde.

De sort methode wordt in de LensBox klasse in de volgende methoden gebruikt:

* changeIndexOfPropertyLabel(PropertyLabel propertyLabel)
* getPropertyLabels()

De PropertyLabelIndexComparator() wordt gebruikt om de propertylabel array op de  
index waarde te sorteren.

De index kan gewijzigd worden door :

* verslepen van de property.
* het toevoegen van een property.

Welke methode gebruiken de propertyindex variabele:

* positionPropertyLabels()
* changeIndexOfPropertyLabel(PropertyLabel propertyLabel)

De propertylabel listarray wordt gevuld in de LensBox klasse met de methode  
addPropertyLabel(). Deze methode roept de methode positionPropertyLabels aan die de propertylabels in de array op het scherm zet op basis van de index waarde. De index waarde bepaald de hoogte waarop de label geplaatst wordt. De index is opgeslagen in het propertybinding object.

## Alfabetische sortering implementeren.

Op basis van de label, default krijgt de label van een property als waarde de naam zonder prefix.  
Deze sortering kan geimplementeerd worden door een Comparator te definieren voor de alfabetische volgorde. Deze volgorde zit in Java String klasse ingebouwd.

## Heuristic sortering implementeren.

Er dient een klasse gedefinieerd te worden die de bucket structuur opbouwt. De LensBox is de eigenaar van deze structuur. Er dient een comparator gedefinieerd te worden die op basis van deze structuur de relatie van twee labels of twee namen van de resources bepaalt.

De bucket structuur bestaat uit een array met 7 elementen die de buckets voorstellen. Elk element bestaat ook weer uit een array die de inbucket volgorde voorstelt. De waarde van het element van de tweede array is de naam of de label van de resource. Het is dus een string array.

De lengte van de inbucket array is niet van te voren bekend en hangt af van de klasse hiërarchie waarin de lens zich bevindt.

Uit deze bucket structuur dient een mapping gedestilleerd te worden die het mogelijk maakt om snel de naam of label van de resource te koppelen aan de volgorde.

De klasse heeft dan een methode getOrder(propertyname) die de ordening als getal terug geeft.  
Voor dat getal kiezen we een float type. Het getal voor de komma geeft de bucket aan, de getallen achter de komma geven de inbucket ordering aan. Alle buckets hebben min. 3 niveaus behalve bucket C. Het aantal niveaus hangt daarbij af van de klassehiërarchie.   
bv:

1.1 bucket A level 1

1.2 bucket A level 2

1.3 bucket A level 3

5.1 bucket E level 1

3.05 bucket C level 5 van de 100

3.75 bucket C level 75 van de 100

## Klasse opzet

De klasse LensBox krijgt een methode ‘sortPropertyLabelHeuristic()’.  
Een nieuwe klasse ‘PropertyLabelHeuristicComparator’, deze klasse wordt in de klasse LensDiagram geinstantieerd en geinitialiseerd voor een LensBox. Deze instantie wordt dan meegegeven aan de LensBox instantie, door een constructor of door een setter.

De klasse ‘PropertyLabelHeuristicComparator()’ bevat de bucketstructuur zoals gedefinieerd door Falco Paul, de klasse ‘FalcoPaulHeuristicSortBucket’

De klasse ‘PropertyLabelHeuristicComparator()’ implementeert de methode compare().  
Deze methode maakt gebruik van de bucketstructuur klasse methode getOrder(property) om een floating getal te krijgen die de ordening aangeeft.

De klasse LensDiagram krijgt een private methode om de FPHSBucket te initialiseren voor de properties van een LensBox.

### Informatie benodigdheden voor de heuristic

1. Label van de property.
2. Prefix van de propertynaam.
3. Uri van de property.
4. Het XSD datatype van een dataproperty.
5. De range rank van een property.
6. Is een property functional of inverse functional.
7. Label/property frequentie.

### Range rank berekening

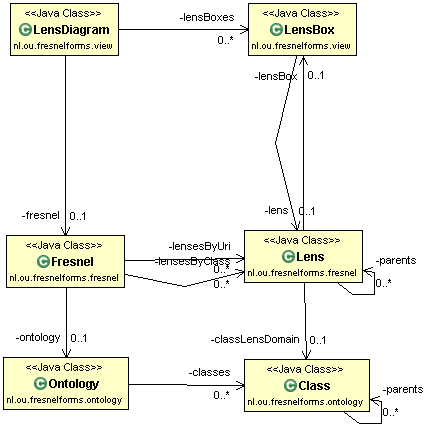
Wordt berekend op basis van de range van een property.  
De dataproperties met xsd type krijgen een rank van 0.  
Van de object properties wordt het gebruik binnen de ontologie bepaalt.

Er wordt een rangeclass frequentie lijst opgesteld door te bepalen hoe vaak een klasse voorkomt in een range definitie van een property. Deze bepaling wordt gedaan voor alle properties in de ontologie.   
Deze lijst wordt aflopend gesorteerd dus populaire klassen eerst.

Per property wordt een hierarchielijst samengesteld van de klasse waarnaar verwezen wordt. Deze hierarchielijst bevat dus alle superklassen van de range klasse. (niet duidelijk is of de range klasse zelf ook meegenomen wordt in deze lijst.)

Om de rank range vast te stellen wordt de rangeclass frequentielijst afgelopen van meest populair naar minder populair. Voor elke klasse wordt gecontroleerd of deze voorkomt in de hiërarchie lijst als dat zo is is de range rank gelijk aan de index in de rangefrequentielijst.

Classe diagram ontology\_hierarchie.gif



Figuur : Klasse hiërachie Ontology, Fresnel en LensDiagram

Bij initialisatie van de lensboxen in het lensdiagram wordt de heuristics informatie doorgegeven aan de constructor van de LensBox. Zodat de lensbox de sortering kan uitvoeren.

### Inventarisatie van de gegevens.

Overzicht van de aanwezigheid van de gegevens die nodig zijn voor de heuristic sortering.

1. De labels worden niet uit de bronontologie uitgelezen maar worden geïnitialiseerd op de naam van de property.
2. De prefix van de property URI is niet bekend. De URI van de property is opgeslagen in de Resource klasse die die zou de prefix terug kunnen geven dmv de URI klasse en de getScheme() methode.
3. De URI van de property is bekend en is opgeslagen in de property klasse.
4. Het XSD datatype is niet bekend en zou verkregen kunnen worden door de datatype property klasse.  
   dmv. Het dataRangetype die een owldatatype object terug kan geven. Dit object kan een XSD enumeration object terug geven.
5. De rangeclassfrequentielijst is niet bekend en zou opgebouwd moeten worden in de ontology klasse.  
   De rangeklasse hiërarchie is bekend. Elke klasse heeft een parents array.
6. Het functional of inverse functional zijn van een property is niet bekend van een property.
7. De property frequentie lijst is nog onbekend en moet door de ontology klasse bepaald worden.

### Implementatiewijzigingen

1. De labels worden uit de bronontologie gelezen en aan het property object toegekend in de OWLImport methode met behulp van de findPropertyLabel methode.
2. In de klasse resource is de String uri vervangen door de klasse URI. Er is een methode toegevoegd getPrefix die het schema gedeelte van de URI terug geeft. Dus van foaf:name wordt foaf terug gegeven.
3. De methode getDatatype toegevoegd aan de dataproperty klasse.
4. Properties functional en inversefunctional toegevoegd aan de property klasse. De owldataproperty klasse heeft wel een isfunctional methode maar geen isInverseFunctional methode. De owlObjectProperty ondersteunt beide methoden. Default zijn beide properties false.

.

# References

Paul, F. (2014). *PROPERTY RANKING APPROACHES FOR SEMANTIC WEB BROWSERS.*