Earth Science Ontology Repository (ESOR) APIs

Table of Contents

Introduction:

The Earth Science Ontology Repository (ESOR) portal is similar to the BioPortal and it focuses on the Earth Science domain. The implementation of ESOR is not based only on keyword search but also involves semantic search.

ESOR can be used as the backend knowledge base for multiple applications. Ex: semi-automatic or automatic entity matching. By inputting related keywords to find the appropriate ontologies to describe Earth Science terms and by analyzing the headings of tabular data, related ontologies are automatically identified.

We use an entity linking system, called Linkipedia and it links the concept mention from textual document to entities on the Web of Data. The Linkipedia framework consists of three standalone web services, each for entity linking, entity search and entity checking.

There are three main steps involved in this process. They are

- Preprocessing
- Computation of weights
- Building the knowledge base and starting servlet.

Preprocessing depends on the data and purpose. It includes merging the data into a single file, sorting this file by subject, obtaining labels from the URL, adding Inverse Property and lexical data clean up. These functionalities are handled by the Preprocessor API.

To compute the weights, we sort the file according to the property, then compute property entropy. The file is then sorted by object and we compute the entity entropy weights. These functionalities are handled by the Preprocessor API.

To build the knowledge base, we require surface form file and find the index surface form and index entity. IndexData API and ReadIndex API are involved in this step.

To perform entity linking, entity search and entity checking, we require Searcher API. Before performing search, linking or checking, the servlets are to be started and this is handled by the EntityServletContext API.

Below are the service interfaces that are exposed through the Earth Science Ontology Repository REST interface, to support Linkipedia search and linking.

The following table provides a list of API methods exposed by ESOR.

Tier:	The tier in which a method is grouped. ***Tier 1:*** Public read, no authentication or access control of content. No content can be created through the DataONE service interfaces. The node cannot act as a replication target. ***Tier 2:*** Read and resolve with access control support. ***Tier 3:*** Write (create, update, delete), possibly limited support for data types.
Version:	Version of API method is available. The lowest version number indicates when the method was added. A version number in parentheses indicates the method is available in that version and is unchanged from the previous version. If more than one version number is present, then the method signature or functionality has changed between API versions. e.g. "1.0, 2.0" indicates that the method was first introduced in Version 1.0 and has been modified in Version 2.0.
HTTP Method:	The HTTP method and path relative to the Base URL. Parameters specified in the URL are indicated by braces. Note that parameters included in a path MUST be properly path encoded, and parameters included as key, value pairs MUST also be properly encoded.
Function:	The function name, associated with an API grouping.
Function Parameters:	Indicates the parameters used when calling the function and the return type of the function.

Methods for ESOR component:

Tier	Version	HTTP Method	Function	Function Parameters
Tier 3	1.0	GET /search?query=…	Searcher.SurfaceFormSearch()	(index) ->void
Tier 3	1.0	GET /search?query=…	Searcher.EntitySearch()	(index) -> void
Tier 3	1.0	GET /linking?query=…	Searcher.EntityLinking()	(Index) ->void
Tier 3	1.0	GET /read?url=...	ReadIndex.readIndex()	(Index) ->void

Tier 3	1.0	NA	IndexData.indexSurfaceForm()	(inputfile, outputfile, entityweightfile, propertyweightfile) ->void
Tier 3	1.0	NA	IndexData.indexEntity()	(inputfile, outputfile, surfaceFormIndex, propertyWeight) ->void
Tier 3	1.0	NA	Preprocessor.preprocessAll()	(inputfile, outputUnsort, surfaceFormPropertiesFile) -> void
Tier 3	1.0	NA	Preprocessor.addInverseProperty()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.removeChars()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.getLabelsFromURL()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.computeEntityWeights()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.computeEntityEntropyWeights()	(inputfile, outputfile, weightFile) -> void
Tier 3	1.0	NA	Preprocessor.computePropertyEntropy()	(inputfile, outputfile) -> void
Tier 3	1.0	NA	Preprocessor.processValueRestriction()	(inputfile, outputfile) -> void
Tier 3	1.0	NA	EntityServletContext.startServlet()	(index, surfaceFormIndex, port) ->void
Tier 3	1.0	NA	Preprocessor.getSurfaceFormData()	(inputfile, surfaceFormPorperties, outputUnsort) ->void
Tier 3	1.0	NA	IndexData.updateEntityIndex()	(inputDir, entityIndex, surfaceFormIndex, propertyWeight, entityWeight) ->void

1. Searcher API

The *Searcher* API implements the methods to perform surface form search, entity search and entity linking. The request is sent in the form of query and the result is obtained in JSON.

Functions defined in *Searcher*:

Tier	Version	HTTP Method	Function	Function Parameters
Tier 3	1.0	GET /search?query=…	Searcher.SurfaceFormSearch()	(index) ->void
Tier 3	1.0	GET /search?query=…	Searcher.EntitySearch()	(index) ->void
Tier 3	1.0	GET /linking?query=…	Searcher.EntityLinking()	(index) ->void

1.1 Searcher.SurfaceFormSearch(index) -> void

Surface form is the textual representation of concepts. This method looks for the surface form for entities from label, name, URL, abbreviations, etc.

Version :	1.0
REST URL:	GET /search?query=…
HTTP Method Parameters:	{index, query}
HTTP Method Returns:	JSON File
Function Parameters:	**index**(*string*) – Surface form index where we can search from.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: IOException e1 e1.printStackTrace()
Example:	http://127.0.0.1:9100/search?query=water

	{"query": "water","time": "0.0","num_result": "20","results": [{"url": "<http://sweet.jpl.nasa.gov/2.3/matrCompound.owl#Water>","score": "0.5008355"}, {"url": "<http://purl.obolibrary.org/obo/ENVO_00002006>","score":"0.24524158"}, {"url": "_:B363a707aX3A14db5a0263aX3AX2D71ed","score": "0.1881693"}, {"url": "<http://purl.obolibrary.org/obo/ENVO_00000489>","score": "0.1881693"},{"url": "<http://purl.obolibrary.org/obo/ENVO_0010149>","score": "0.13305578"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenOceanCoastal.owl#OceanTide>","score": "0.13305578"},{"url": "<http://purl.obolibrary.org/obo/ENVO_00000063>","score": "0.11212227"},{"url": "<http://sweet.jpl.nasa.gov/2.3/matrWater.owl#BrackishWater>","score": "0.10991287"},{"url": "<http://sweet.jpl.nasa.gov/2.3/realmHydro.owl#WaterTable>","score": "0.10991287"},{"url": "<http://purl.obolibrary.org/obo/ENVO_01000108>","score": "0.10746904"},{"url": "<http://purl.obolibrary.org/obo/ENVO_01000109>","score": "0.10746904"},{"url": "<http://purl.obolibrary.org/obo/ENVO_01000266>","score": "0.10746904"},{"url": "<http://purl.obolibrary.org/obo/ENVO_01000268>","score": "0.10746904"},{"url": "<http://purl.obolibrary.org/obo/ENVO_00000549>","score": "0.10469515"},{"url": "<http://purl.obolibrary.org/obo/ENVO_00002227>","score": "0.10469515"},{"url": "<http://purl.obolibrary.org/obo/ENVO_02000049>","score": "0.10469515"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenAtmoPrecipitation.owl#WaterSpout>","score": "0.10469515"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenEnvirImpact.owl#SaltWaterIntrusion>","score": "0.10469515"},{"url": "<http://purl.obolibrary.org/obo/ENVO_00000061>","score": "0.101404816"},{"url": "<http://purl.obolibrary.org/obo/ENVO_00000570>","score": "0.101404816"}]}

1.2 Searcher.EntitySearch(index) -> void

This method searches for the concept mentions that provide the context information. Searches only the first concept mention.

It converts the contents of the query into lower case, gets the terms by splitting the contents based on “,” and retrieves the URL for each of those terms mentioned in the query.

In the query, all concept mentions are separated by ",". Types of concept mentions are described by "()" and it accepts multiple types separated by ";". Ex: Oxford(Place;University)

Weights for each term can be passed using "|". This applies only for searching. Ex: Oxford(Place;University)|40, London|30

Version :	1.0
REST URL:	GET /search?query=…

HTTP Method Parameters:	{index, query}
HTTP Method Returns:	JSON File
Function Parameters:	**index**(*string*) – Index where we can search from. Sorted by Subject.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: IOException e1 e1.printStackTrace()
Example:	http://127.0.0.1:9100/search?query=cell,mamatele,karif,levante,kona {"query": "cell,mamatele,karif,levante,kona","time": "0.0","num_result": "10","results": [{"url": "<http://sweet.jpl.nasa.gov/2.3/phenFluidDynamics.owl#Cell>","score": "0.16294761"},{"url": "<http://sweet.jpl.nasa.gov/2.3/reprDataModel.owl#Cell>","score": "0.16225637"},{"url": "<http://purl.obolibrary.org/obo/ENVO_02000008>","score": "0.042874437"},{"url": "<http://sweet.jpl.nasa.gov/2.3/matrEnergy.owl#FuelCell>","score": "0.042874437"},{"url": "<http://sweet.jpl.nasa.gov/2.3/matrEnergy.owl#PhotovoltaicCell>","score": "0.042874437"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenAtmoPressure.owl#ConvectionCell>","score": "0.042874437"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenAtmoWind.owl#WalkerCell>","score": "0.042874437"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenFluidDynamics.owl#CellCluster>","score": "0.042874437"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenOceanDynamics.owl#ColdCoreCell>","score": "0.042874437"},{"url": "<http://sweet.jpl.nasa.gov/2.3/phenOceanDynamics.owl#WarmCoreCell>","score": "0.042874437"}]}

1.3 Searcher.EntityLinking(index) -> void

This method links all concept mentions in the query to the knowledge base. Searches all the concept mentions.

It obtains the terms from the noun phrases in the query and retrieves the URL for each of those terms mentioned in the query.

In the query, all concept mentions are separated by ",". Types of concept mentions are described by "()" and it accepts multiple types separated by ";". Ex: Oxford(Place;University)

Version :	1.0
REST URL:	GET /linking?query=…
HTTP Method Parameters:	{index, query}
HTTP Method Returns:	JSON File
Function Parameters:	**index**(*string*) – Index where we can search from. Sorted by Subject.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: IOException e1 e1.printStackTrace()
Example:	http://127.0.0.1:9100/linking?query=cell,mamatele,karif,levante,kona {"query": "cell,mamatele,karif,levante,kona","time": "4997","results": [{"entity_mention": "cell","annotations": [{"url": "<http://sweet.jpl.nasa.gov/2.3/phenFluidDynamics.owl#Cell>","score": "0.44081637"}]},{"entity_mention": "mamatele","annotations": [{"url": "<http://sweet.jpl.nasa.gov/2.3/phenAtmoWindMesoscale.owl#Mamatele>","score": "0.56697744"}]},{"entity_mention": "karif","annotations": [{"url": "<http://sweet.jpl.nasa.gov/2.3/phenAtmoWindMesoscale.owl#Karif>","score": "0.564228"}]},{"entity_mention": "levante","annotations": [{"url": "<http://sweet.jpl.nasa.gov/2.3/phenAtmoWindMesoscale.owl#Levante>","score": "0.564228"}]},{"entity_mention": "kona","annotations": [{"url": "<http://sweet.jpl.nasa.gov/2.3/phenAtmoWindMesoscale.owl#Kona>","score": "0.59529567"}]}]}

2. Read Index API

The *ReadIndex* API implements a method that reads the knowledge base and looks up what is indexed, for entity checking.

Functions defined in *ReadIndex:*

Tier	Version	HTTP Method	Function	Function Parameters
Tier 3	1.0	GET /read?url=...	ReadIndex.readIndex()	(index) ->void

2.1 ReadIndex.readIndex(index) -> void

This method is used in entity checking and it reads the knowledge base to see what is indexed.

Version :	1.0
REST URL:	GET /read?url=...
HTTP Method Parameters:	{index, url}
HTTP Method Returns:	JSON File
Function Parameters:	**index**(*string*) – Knowledge index. The knowledge base for linking and searching.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.printStackTrace()
Example:	http://127.0.0.1:9100/read?url=http://sweet.jpl.nasa.gov/2.3/phenFluidDynamics.owl#Cell {"query": "<http://sweet.jpl.nasa.gov/2.3/phenFluidDynamics.owl>","triples": [{"triple": "<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Ontology>"},{"triple": "<http://tool.eal.org/urlName> phen Fluid Dynamics owl"},{"triple": "<http://www.w3.org/2000/01/rdf-schema#label> SWEET Ontology"},{"triple": "<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/phen.owl>"},{"triple": "<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/procPhysical.owl>"},{"triple": "<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/propPressure.owl>"},{"triple": "<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/rela.owl>"},{"triple": "<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/relaSci.owl>"},{"triple":

	"<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl>"},{"triple": "<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/reprSpaceGeometry.owl>"},{"triple": "<http://www.w3.org/2002/07/owl#imports> <http://sweet.jpl.nasa.gov/2.3/statePhysical.owl>"},{"triple": "<http://www.w3.org/2002/07/owl#versionInfo> 2.3"}]}

3. Entity Servlet Context Class

The *EntityServletContext* class implements a method that starts the servlet.

Functions defined in *EntityServletContext:*

Tier	Version	HTTP Method	Function	Function Parameters
Tier 3	1.0	NA	EntityServletContext.startServlet()	(index, surfaceFormIndex, port) ->void

3.1 EntityServletContext.startServlet(index, surfaceFormIndex, port) -> void

This method is used to start the servlets. Once the servlets are started, we can perform linking or searching.

Version :	1.0
Function Parameters:	**index**(*string*) – Knowledge index. The knowledge base for linking and searching. **surfaceFormIndex**(*string*) – Surface form index, built from the processed N-triple file containing only surface form information. **port** (*int*) – default 8080.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.printStackTrace()
Example:	java -jar Linkipedia.jar startServlet knowledgeIndex surfaceFormIndex port(optional)

4. Index Data Class

The *IndexData* class implements the methods to create index for surface form and entity, and to update the entity index.

Functions defined in *IndexData:*

Tier	Version	HTTP Method	Function	Function Parameters
Tier 3	1.0	NA	IndexData.indexSurfaceForm()	(inputfile, outputfile, entityweightfile, propertyweightfile) ->void
Tier 3	1.0	NA	IndexData.indexEntity()	(inputfile, outputfile, surfaceFormIndex, propertyWeight) ->void
Tier 3	1.0	NA	IndexData.updateEntityIndex()	(inputDir, entityIndex, surfaceFormIndex, propertyWeight, entityWeight) ->void

4.1 IndexData.indexSurfaceForm(inputfile, outputfile, entityweightfile, propertyweightfile) -> void

This method creates the surface form index used for searching.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – Processed N-triple file, sorted by subject, containing only surface form information. **outputfile**(*string*) – File containing surface form index we can search from. **entityweightfile**(*string*) – Entropy based ranking for entities. Output of computeEntityEntropyWeights. **propertyweightfile**(*string*) – Information entropy for properties. Output of computePropertyEntropy.
Returns:	Returns nothing.
Return type:	void
Raises:	No Exception.

Example:	java -jar Linkipedia.jar indexSurfaceForm inputfile outputIndexName entityEntropyFile propertyEntropyFile

4.2 IndexData.indexEntity(inputfile, outputfile, surfaceFormIndex, propertyWeight) -> void

This method creates the entity index and gives the final knowledge base that can be used for linking and searching.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – All entities in N-triple format sorted by subject. **outputfile**(*string*) – Final knowledge base for linking and searching purpose. **surfaceFormIndex**(*string*) – Surface form index where we can search from. **propertyweight**(*string*) – Information entropy for properties. Output of computePropertyEntropy.
Returns:	Returns nothing.
Return type:	void
Raises:	No Exception.
Example:	java -jar Linkipedia.jar indexEntity inputfile outputfile surfaceFormIndex propertyEntropyWeight

4.3 IndexData.updateEntityIndex(inputDir, entityIndex, surfaceFormIndex, propertyWeight, entityWeight) -> void

This method updates the entity index.

Version :	1.0
Function Parameters:	**inputDir**(*string*) – Input file with all entities in N-triple format sorted by subject. **entityIndex**(*string*) – Final knowledge base for linking and searching purpose. **surfaceFormIndex**(*string*) – Surface form index where we can search from. **propertyWeight**(*string*) – Information entropy for properties. Output of computePropertyEntropy. **entityWeight**(*string*) – Entropy based ranking for entities. Output of computeEntityEntropyWeight.

Returns:	Returns nothing.
Return type:	void
Raises:	No Exception.

5. Preprocessor Class

The *Preprocessor* class contains the methods to preprocess the data and eliminates characters that the user will not search for (lexical data clean up), methods for merging the data into a single file, sorting the file by subject, by property and by object, obtaining labels from URL, adding Inverse Property, computing entity weights, computing entity entropy weights, computing property entropy, processing value restrictions and obtaining surface form data.

Functions defined in *Preprocessor:*

Tier	Version	HTTP Method	Function	Function Parameters
Tier 3	1.0	NA	Preprocessor.preprocessAll()	(inputfile, outputUnsort, surfaceFormPropertiesFile) -> void
Tier 3	1.0	NA	Preprocessor.addInverseProperty()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.removeChars()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.getLabelsFromURL()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.computeEntityWeights()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.computeEntityEntropyWeights()	(inputfile, outputfile, weightFile) ->void
Tier 3	1.0	NA	Preprocessor.computePropertyEntropy()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.processValueRestriction()	(inputfile, outputfile) ->void
Tier 3	1.0	NA	Preprocessor.getSurfaceFormData()	(inputDir, entityIndex, surfaceFormIndex,

				propertyWeight, entityWeight) ->void

5.1 Preprocessor.preprocessAll(inputfile, outputUnsort, surfaceFormPropertiesFile) -> void

This method obtains surface form property and performs content processing.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – Ontologies in rdf or owl or ttl formats. **outputUnsort**(*string*) – Unsorted output file. **surfaceFormPropertiesFile**(*string*) – Surface form property.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.printStackTrace()
Example:	cat *.nt >> merged.nt

5.2 Preprocessor.addInverseProperty(inputfile, outputfile) -> void

This method takes as input an N-triple file and deletes “>” in the object and adds the inverse property, creating a new triple.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – N-triple file. **outputfile**(*string*) – N-triple file with inverse property added.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.getMessage()
Example:	java -jar Linkipedia.jar addInverseProperty merged_sort1_v1 merged_sort1_v2

5.3 Preprocessor.removeChars(inputfile, outputfile) -> void

This method does the lexical data clean up. It removes all the characters except alphabets and numbers. It deletes the '^^' sign and the content after that, then deletes '@' and the content after that, deletes double quotes, single quotes and the period at the end of lines. The subject with double quotes will not be included in the new triple.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – N-triple file. **outputfile**(*string*) – N-triple file with junk characters removed.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.getMessage()
Example:	java -jar Linkipedia.jar removeChars merged_sort1_v2 merged_sort1_v3 **Delete ^^ and the content after that:** sed "s/\\^\\^.*//g" merged_sort1_v3.nt >merged_sort1_v3_t1.nt **Delete '@' and the content after that:** sed "s/@.*//g" merged_sort1_v3_t1.nt > merged_sort1_v3_t2.nt printf '%s\0' *.nt | xargs -0 cat > merged_sort1_v3_t2.txt **Delete double quote:** :%s/\v"//gc **Delete period at the end of the lines:** :%s/\v(.+)\.$/\1/gc **Delete single quote:** :%s/\v'/ /gc :%s/\v((http)@<!:(.*)@=)&(:([^>]*$)@=)/ /gc

5.4 Preprocessor.getLabelsFromURL(inputfile, outputfile) -> void

This method generates surface form from URL. It converts the URL names to phrases to obtain surface form.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – N-triple file, sorted by subject. **outputfile**(*string*) – File with surface forms.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.getMessage()
Example:	java -jar Linkipedia.jar getLabelsFromURL merged_sort1 merged_sort1_v1 **Sample Output:** “George Washington” for dbpedia:George_Washington

5.5 Preprocessor.computeEntityWeights(inputfile, outputfile) -> void

This method is used to obtain weights for the list of objects that other entities are pointed to.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – File with a list of objects that other entities are pointed to. **outputfile**(*string*) – File with weights for each object.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.printStackTrace()

5.6 Preprocessor.computeEntityEntropyWeights(inputfile, outputfile, weightFile) -> void

This method computes entropy based rankings for entities. Takes two input files, one of which is an n-triple file sorted by object and the other is a property weight file, and computes weights for each entity. It reads the property entropy and the first triple where the object is not a string. It then computes the entropy and writes it to a file.

Version :	1.0

Function Parameters:	**inputfile**(*string*) – N-triple file sorted by object. **outputfile**(*string*) – File with weights for entities. **weightFile**(*string*) – Input file with property weight, output of computePropertyEntropy.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.printStackTrace()
Example:	java -jar Linkipedia.jar computeEntityEntropyWeights merged_sort3 entity_entropy property_entropy **Sample Output:** <http://www.w3.org/ns/prov#Activity> 7.110959 <http://www.w3.org/ns/prov#ActivityInfluence> 4.996974 <http://www.w3.org/ns/prov#Agent> 6.580115 <http://www.w3.org/ns/prov#AgentInfluence> 4.996974 <http://www.w3.org/ns/prov#unqualifiedForm_InverseProperty> <http://www.w3.org/ns/prov#Association> 1.0 <http://www.w3.org/ns/prov#unqualifiedForm_InverseProperty> <http://www.w3.org/ns/prov#Attribution> 1.0

5.7 Preprocessor.computePropertyEntropy(inputfile, outputfile) -> void

This method computes the information entropy for properties. It takes as input, an N-triple file which is sorted by property and computes property entropy based on the number of occurrences and the number of objects.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – N-triple file sorted by property. **outputfile**(*string*) – File with property entropy.
Returns:	Returns nothing.
Return type:	void

Raises:	Generic Exception: Exception e e.printStackTrace()
Example:	java -jar Linkipedia.jar computePropertyEntropy merged_sort2 property_entropy **Sample Output:** <http://sweet.jpl.nasa.gov/2.3/relaSci.owl#hasUnit> 14.251937326507834 <http://sweet.jpl.nasa.gov/2.3/relaSci.owl#hasUpperQuantity> 21.72326583694641 <http://sweet.jpl.nasa.gov/2.3/relaSci.owl#kills> 1.0 <http://sweet.jpl.nasa.gov/2.3/relaSci.owl#moreActiveThan> 13.23576805877876 <http://sweet.jpl.nasa.gov/2.3/relaSci.owl#moreExtensiveThan> 21.72326583694641 <http://sweet.jpl.nasa.gov/2.3/relaSci.owl#spawn> 21.72326583694641 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#fartherThan> 1.0 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#greaterVerticalExtentThan> 21.72326583694641 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#hasArea> 21.72326583694641 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#hasAverageDepth> 21.72326583694641 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#hasDirection> 13.360639990153718 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#hasLatitudeBand> 21.723265836946403 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#hasLatitudeLine> 21.72326583694641 <http://sweet.jpl.nasa.gov/2.3/relaSpace.owl#hasMaximumDepth> 21.72326583694641

5.8 Preprocessor.processValueRestriction(inputfile, outputfile) -> void

This method retrieves all the blank node triples and processes them. Checks for the blank node in the subject position as well as in the object position and selects them for processing. Processes owl:restrictions on someValuesFrom and allValuesFrom.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – N-triple file. **outputfile**(*string*) – Processed N-triple file which is sorted.
Returns:	Returns nothing.

Return type:	void
Raises:	Generic Exception: Exception e e.printStackTrace()

5.9 Preprocessor.getSurfaceFormData(inputfile, surfaceFormPorperties, outputUnsort) -> void

This method gets all the surface form for entities. The input file is sorted by subject and includes getLabelFromURL. If the triples in the input file contains any properties from surfaceFormProperties, then the method obtains phrases from URL and the substrings for surface form. It searches triples that contain 'name' or 'label'.

Version :	1.0
Function Parameters:	**inputfile**(*string*) – N-triple file, sorted by subject. **surfaceFormProperties**(*string*) – Properties file. **outputUnsort**(*string*) – Unsorted surface form file.
Returns:	Returns nothing.
Return type:	void
Raises:	Generic Exception: Exception e e.printStackTrace()
Example:	grep -i '.*name.*|.*label.*' merged_sort1_v3>surface_form_file grep surface form data (sorted by subject) -> surface form file