Configuring Named Entity Extraction through Real-Time Exploitation of Linked Data

Pavlos Fafalios, Manolis Baritakis and Yannis Tzitzikas

Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science (ICS)
Information Systems Laboratory (ISL)

University of Crete
Computer Science Department
Greece

Outline

- Introduction
  - *Named entity extraction / Motivation / Contribution*
- The proposed approach
  - *The configuration model*
- The system X-Link
  - *Architecture / Functionality / Configurability*
- Evaluation
  - *User study / Case study*
- Conclusion and Future Research

Named Entity Extraction

- Named Entity Extraction (NEE) is the process of identifying entities in texts and linking them to related semantic resources
- Useful in many applications:
  - *Annotating documents, Question answering, Results post-processing, …*
- The Semantic Web realization highly depends on the availability of metadata (structured content in general) describing Web content
  - A NEE system can automate the extraction of structured data from Web content
- A lot of information about named entities is already available as Linked Open Data (LOD)
  - The exploitation of LOD by a NEE system can bring wide coverage and fresh information

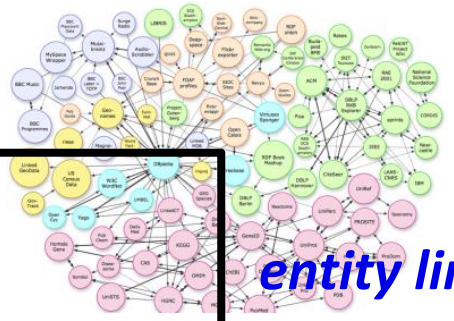
LOD-based Named Entity Extraction

named entity recognition

*(using **NLP/ML** and/or **Gazetteers**)*

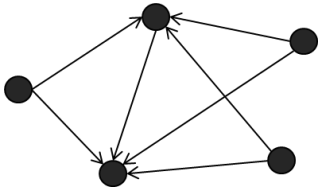
Fish Species

- chum salmon
- coho salmon
- atlantic Salmon
- poacher
- sockeye salmo
- chinook salmon

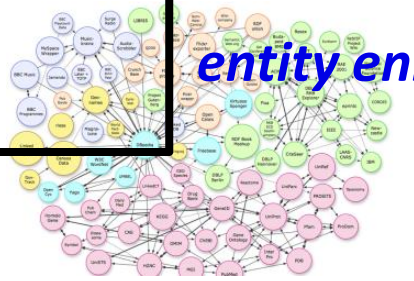


entity linking

http://dbpedia.org/resource/Chum_salmon



kingdom: Animali
phylum: Chordata
class: Actinopterygii
order: Salmoniformes
binomial authority: J. Walbaum



entity enrichment

Motivation (1/2)

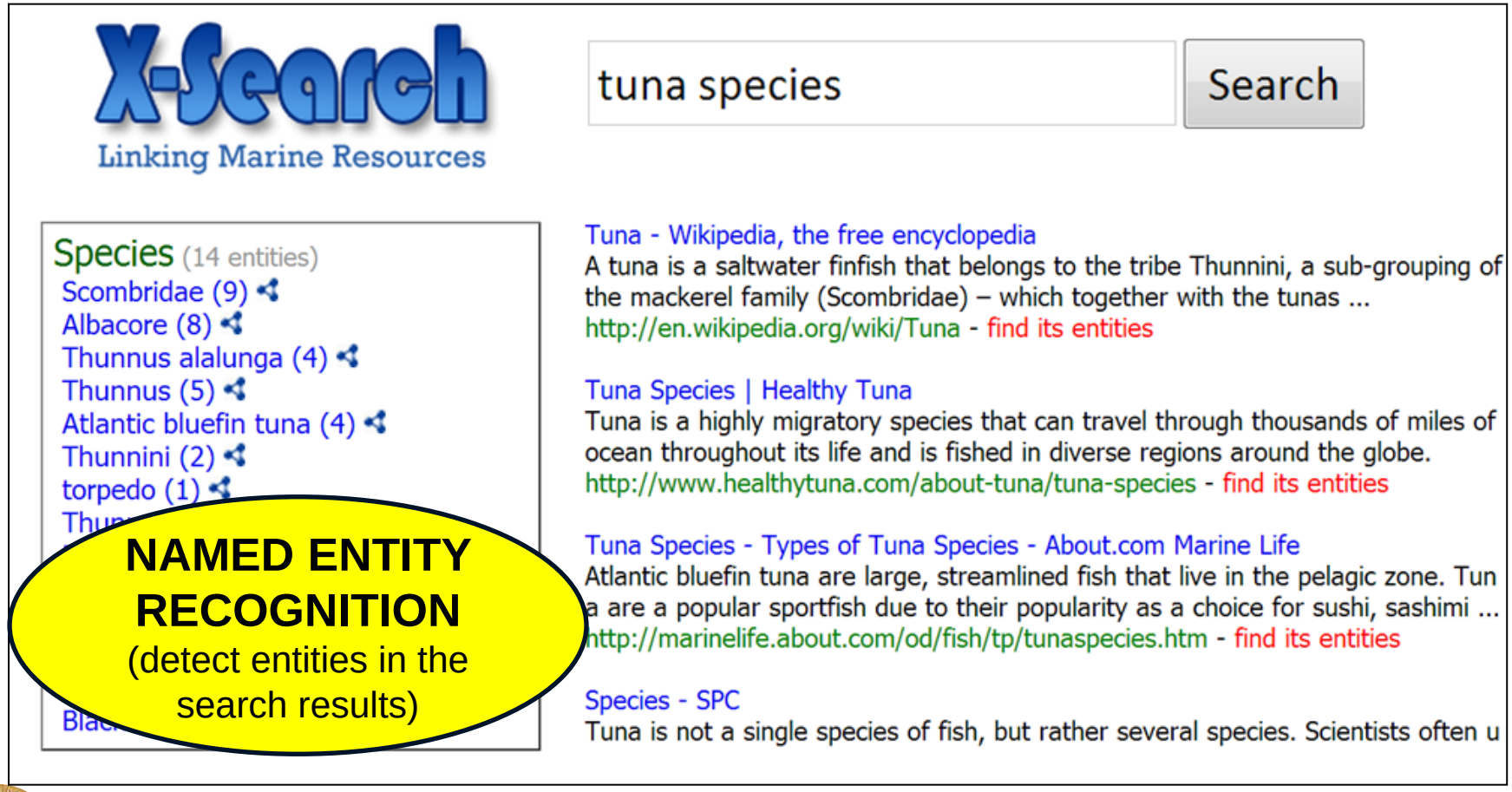
- There are many LOD-based tools that support NEE
  - DBpedia Spotlight, AlchemyAPI, OpenCalais, AIDA, Wikimeta, …
- Configuring an existing NEE system for building domain specific applications is…challenging!
  - Time consuming and laborious even for persons with computer science background
  - requires many technical skills
- Existing tools are mainly dedicated to one specific Knowledge Base which is indexed beforehand
  - thereby, they do not exploit the dynamic and distributed nature of LOD

Motivation (2/2)

- In existing NEE tools, the user (an admin or a developer) cannot easily:
  - define its own interesting types/categories of entities
  - update/extend an existing category with additional entities coming from a new Knowledge Base (KB)
  - Specify how to link the identified entities with semantic resources
  - Control how to enrich the identified entities, i.e. configure the properties that are useful for a particular application
    - e.g. retrieve images, or a description in a specific language
  - Inspect whether and how the identified entities are connected
    - not within the document but as entities in general

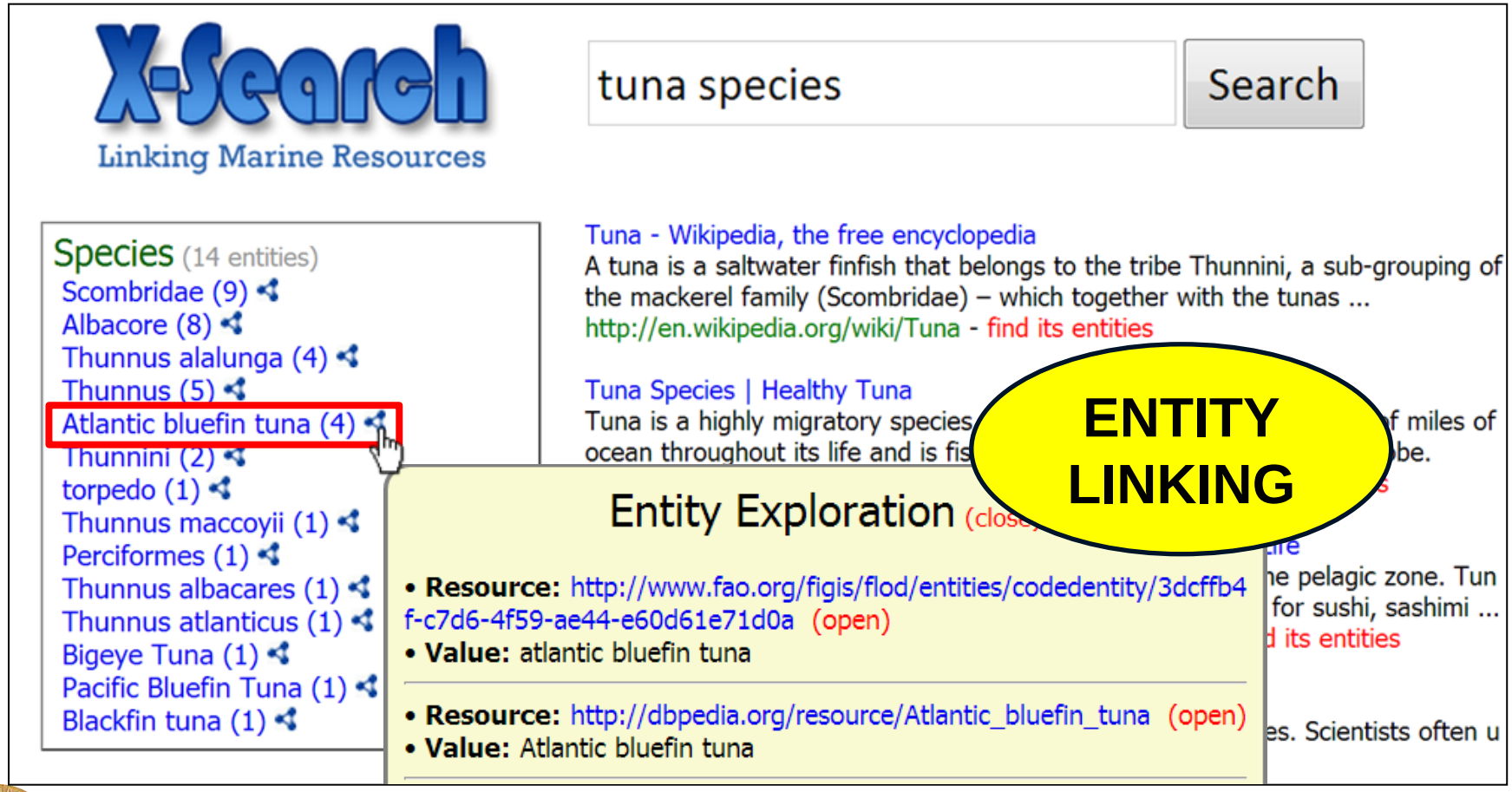
Motivating Example (1/2)

<u>Application:</u> Semantic post-processing of search results



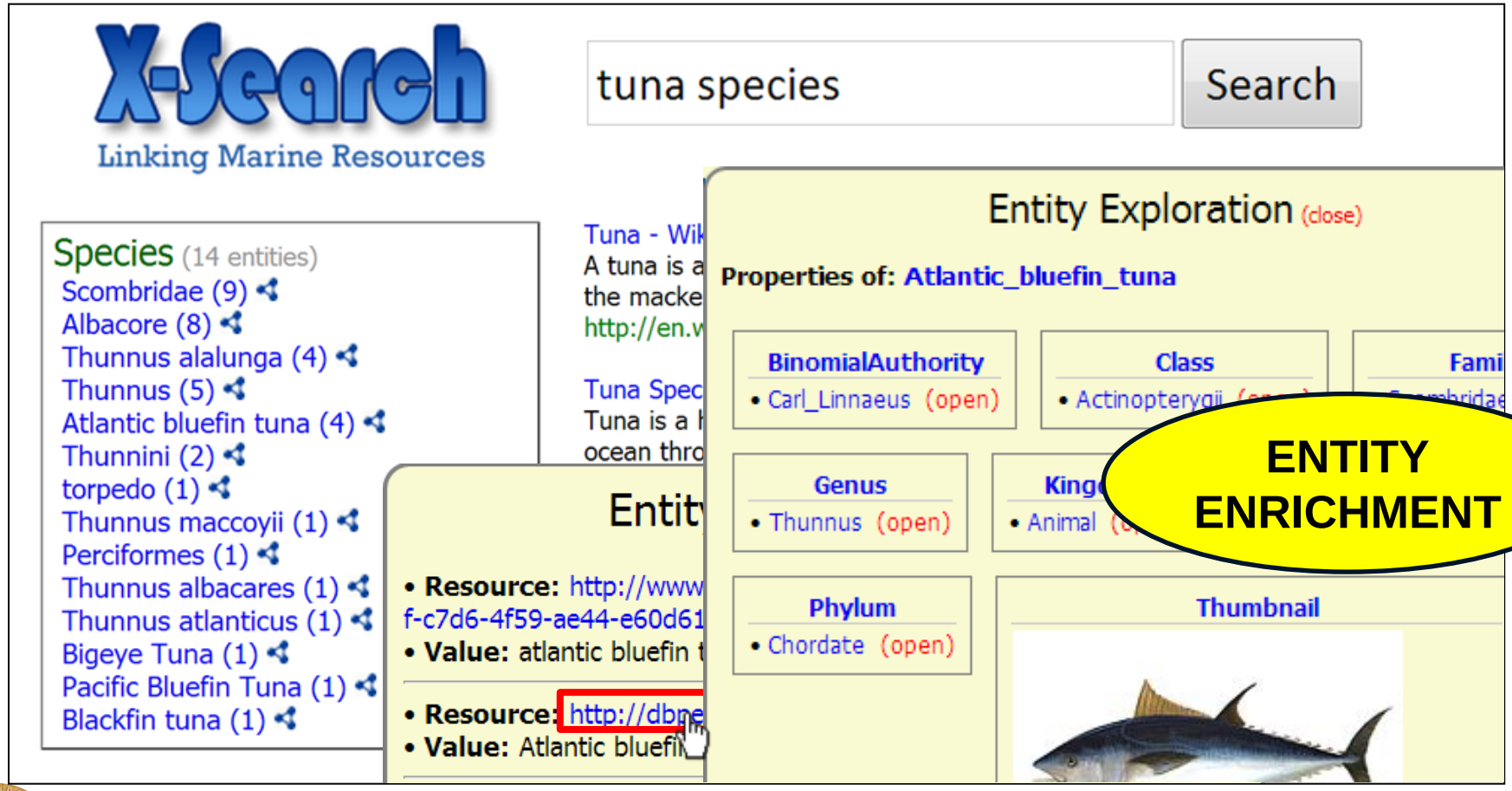
Motivating Example (1/2)

<u>Application:</u> Semantic post-processing of search results



Motivating Example (1/2)

<u>Application:</u> Semantic post-processing of search results



Motivating Example (2/2)

- Each community of users has different needs
  - X-Search should support different configurations
- The needs of a community constantly change
  - We would like to be able to dynamically change the configuration *(at any time, without requiring to redeploy the system)*
- The LOD constantly grows/changes
  - X-Search should be aware of the “fresh” information

Contribution

- We will see:
  - A generic model for configuring (dynamically) a LOD-based NEE system
    - *which can be exploited by existing NEE systems*
  - X-Link, a fully configurable NEE tool that supports the proposed model

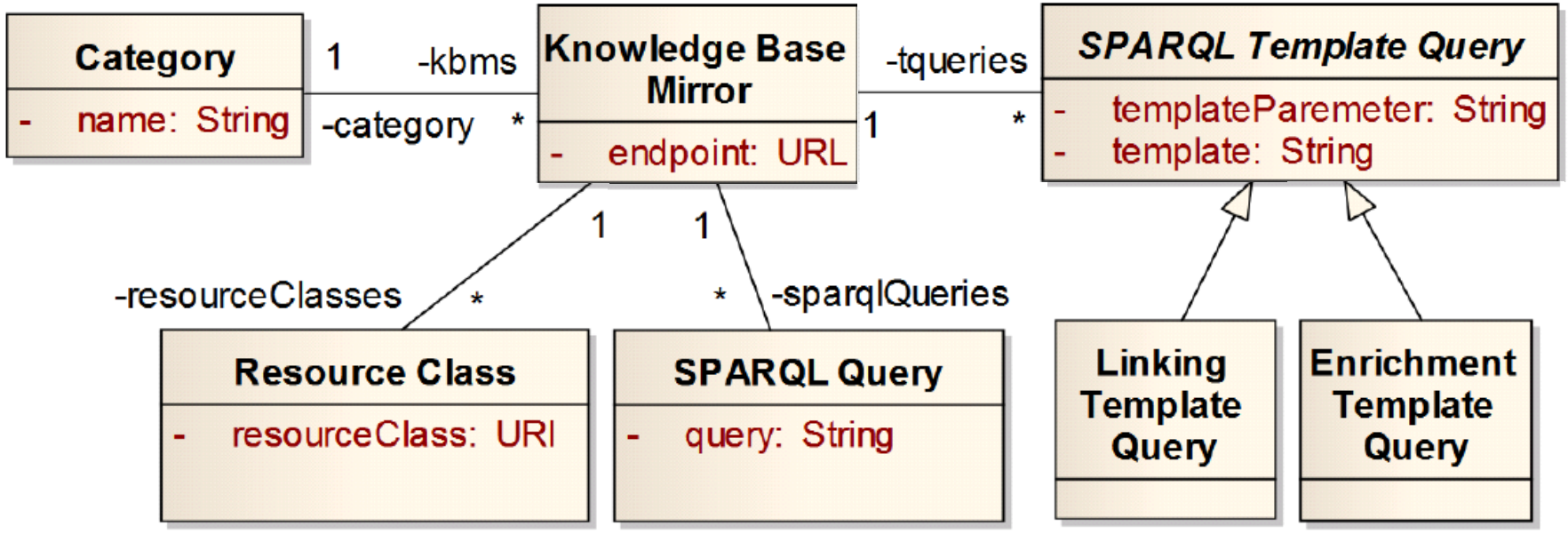
and…

  - The results of a task-based user study
  - The results of a case study
  - Lessons learned, limitations, how to cope with the limitations

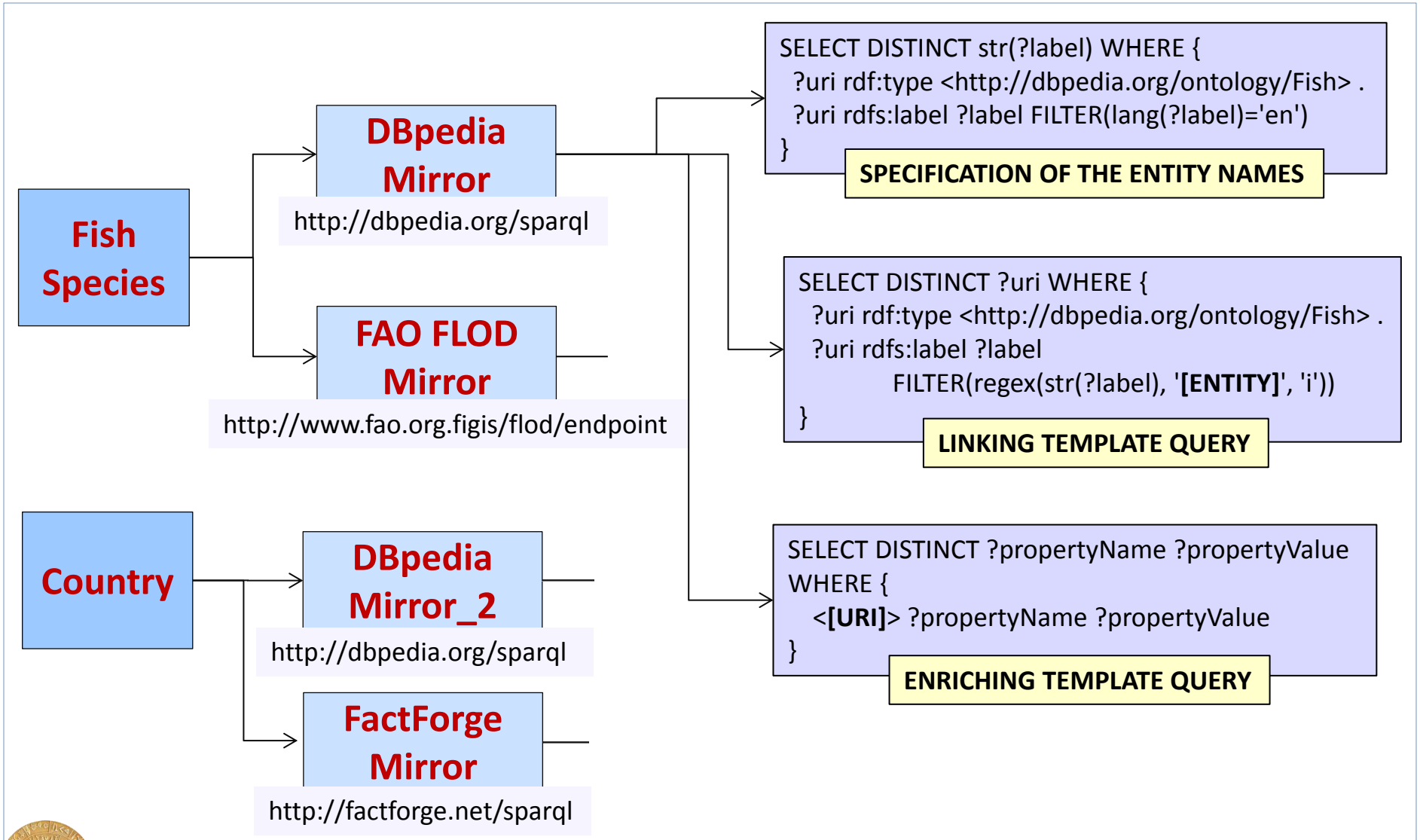
Outline

- Introduction
  - *Named entity extraction / Motivation / Contribution*
- The proposed approach
  - *The configuration model*
- The system X-Link
  - *Architecture / Functionality / Configurability / Applications*
- Evaluation
  - *User study / Case study*
- Conclusion and Future Research

Configuration Model



Example of the Configuration Model



Specification of the entity names of interest - Example

http://dbpedia.org/sparql

```
SELECT DISTINCT str(?label) WHERE {
   ?uri rdf:type <http://dbpedia.org/ontology/Fish> .
   ?uri rdfs:label ?label FILTER(lang(?label)='en')
}
```

Acanthicus
Acanthurus
Acanthurus achilles
Acanthurus albipectoralis
Acanthurus auranticavus
Acanthurus chronixis
....

Linking Template Query – Example

http://dbpedia.org/sparql

```
SELECT DISTINCT ?uri WHERE {
  ?uri rdf:type <http://dbpedia.org/ontology/Fish> .
  ?uri rdfs:label ?label FILTER(regex(str(?label), '[ENTITY]', 'i'))
}
```

Linking Template Query – Example

http://dbpedia.org/sparql

For the entity name "**chum salmon**":

```
SELECT DISTINCT ?uri WHERE {
  ?uri rdf:type <http://dbpedia.org/ontology/Fish> .
  ?uri rdfs:label ?label FILTER(regex(str(?label), 'chum salmon', 'i'))
}
```

http://dbpedia.org/resource/Chum_salmon

Enriching Template Query – Example

http://dbpedia.org/sparql

```
SELECT DISTINCT ?propertyName ?propertyValue
WHERE {
   <[URI]> ?propertyName ?propertyValue
}
```

Enriching Template Query – Example

http://dbpedia.org/sparql

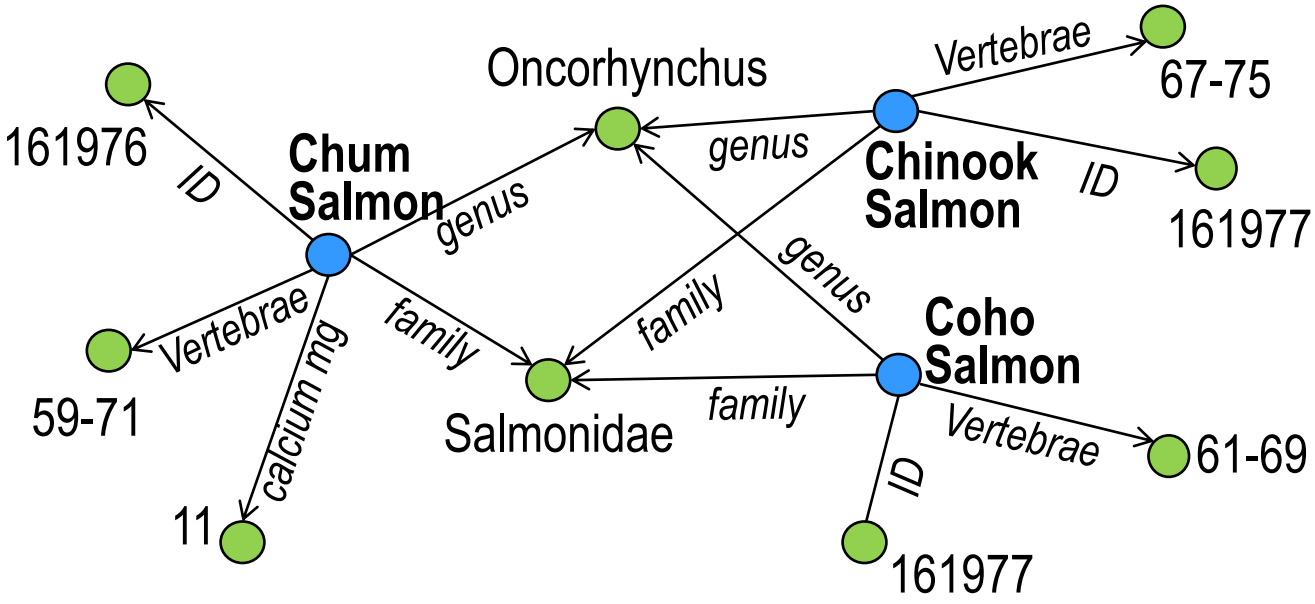
For the entity URI "**http://dbpedia.org/resource/Chum_salmon**":

```
SELECT DISTINCT ?predicate ?object
WHERE {
  <http://dbpedia.org/resource/Chum_salmon> ?predicate  ?object
}
```

http://dbpedia.org/ontology/**binomialAuthority**	http://dbpedia.org/resource/**Johann_Julius_Walbaum**
http://dbpedia.org/ontology/**class**	http://dbpedia.org/resource/**Actinopterygii**
http://dbpedia.org/ontology/**family**	http://dbpedia.org/resource/**Salmonidae**
http://dbpedia.org/ontology/**genus**	http://dbpedia.org/resource/**Oncorhynchus**
http://dbpedia.org/ontology/**kingdom**	http://dbpedia.org/resource/**Animal**
http://dbpedia.org/ontology/**order**	http://dbpedia.org/resource/**Salmonidae**
http://dbpedia.org/ontology/**phylum**	http://dbpedia.org/resource/**Chordate**
… … … …	… … … …

Inferring the connectivity of the identified entities

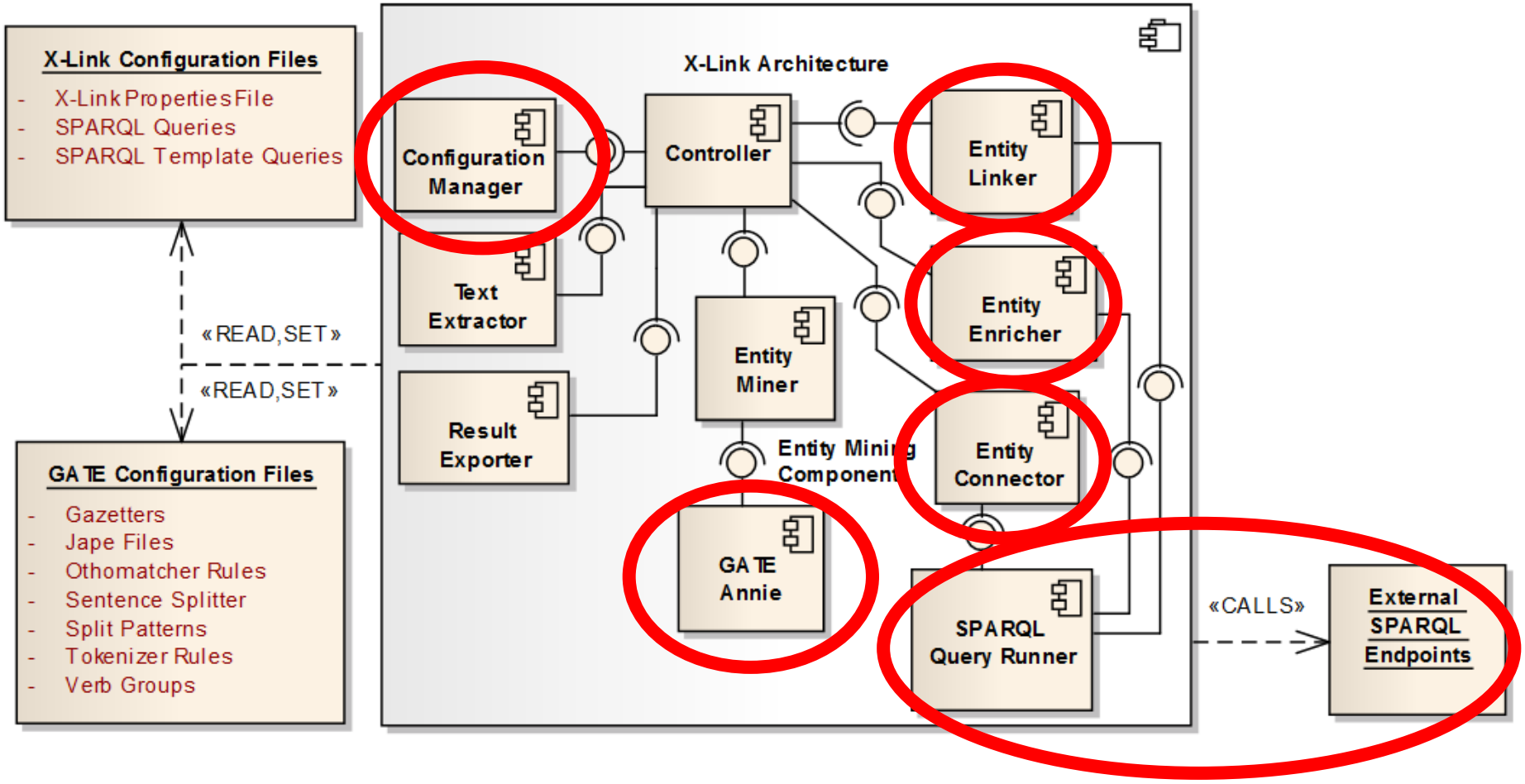
Graph(doc,r)



Outline

- Introduction
  - *Named entity extraction / Motivation / Contribution*
- The proposed approach
  - *The configuration model*
- The system **X-Link**
  - *Architecture / Functionality / Configurability / Applications*
- Evaluation
  - *User study / Case study*
- Conclusion and Future Research

Architecture



Functionality – Input / Output

- Supported file types
  - Plain text
  - HTML
  - PDF
  - .doc, .docx
  - .ppt, .pptx
  - XML-based
- Output
  - Currently, in XML and CSV
  - Soon in RDF (exploiting the Open Annotation standard)

Functionality – Entity Mining and Entity Linking

- Entity Mining
  - Using Gate ANNIE
  - Currently, no disambiguation is applied (when using gazetteers)
    - *If an entity name exists in two supported categories, then this entity is returned twice, one for each category.*
  - Fuzzy matching: identification of an entity that does not match exactly an entity in a category's gazetteer
    - Using configurable edit (Levenshtein) distance that depends on entity name's length
- Entity Linking
  - For a detected entity name, X-Link returns the matching URIs according to the specified template queries

Functionality – Entity Enrichment and Entity Connectivity

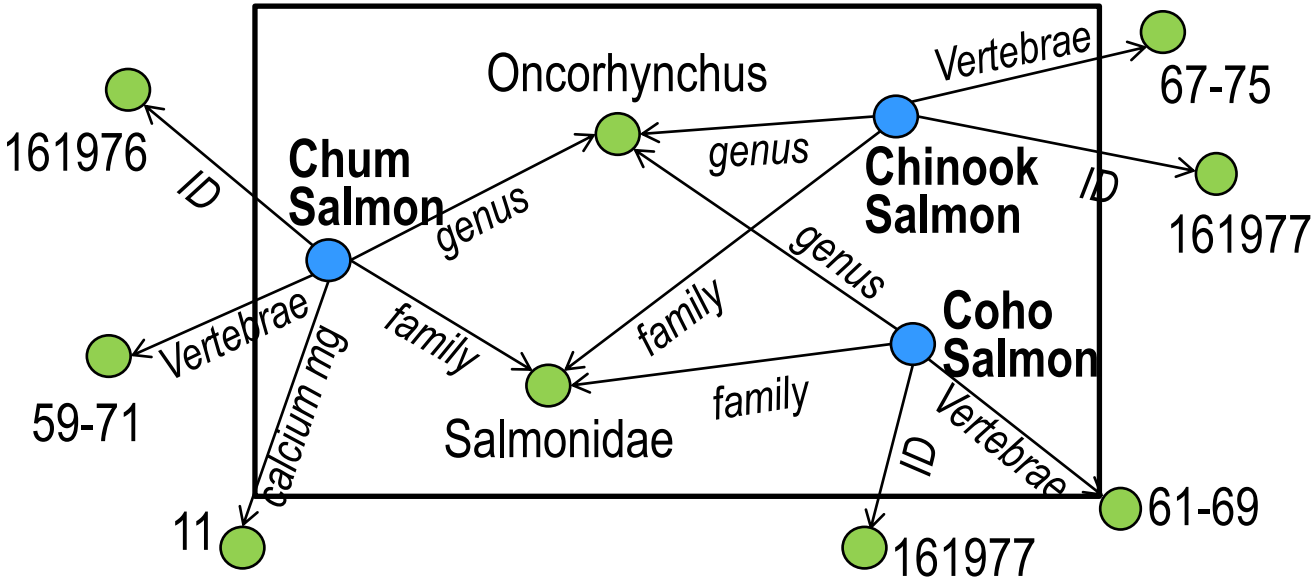
- Entity Enrichment
  a) Retrieve RDF triples
     - According to the specified template queries, or
     - Select one of some predefined (common) types of properties:

       i) outgoing, ii) incoming, iii) both outgoing and incoming,
       iv) outgoing in a language, v) both outgoing in a language and incoming

  b) Inspect the connectivity of the entity URIs
     - Support of `Graph(doc,r)`
       - *Triples describing the identified entities*
     - Support of `ConnectGraph(doc,r)`
       - *Keep only the triples that connect the identified entities*

Inspecting the connectivity of the entity URIs

ConnectGraph(doc,r)



Configurability

- File-based configuration

x-link.properties

```
1   xlink.categories.supported = Fish;Country;Water_Area;Disease;Drug;Protein;Chemical_Substance
2   xlink.categories.active = Fish;Country;Water_Area
3   xlink.categories.Fish.kbms = dbpedia_fish
4   xlink.categories.Fish.kbms.dbpedia_fish.endpoint = http://dbpedia.org/sparql
5   xlink.categories.Fish.kbms.dbpedia_fish.resourceclasses = http://dbpedia.org/ontology/Fish;
                                                              http://umbel.org/umbel/rc/Fish
6   xlink.categories.Fish.kbms.dbpedia_fish.templatequeries.linking = C:/tmpls/dbpFishLinking.sparql
7   xlink.categories.Fish.kbms.dbpedia_fish.templatequeries.linking.parameter = [ENTITY]
8   xlink.categories.Fish.kbms.dbpedia_fish.templatequeries.enriching = C:/tmpls/dbpFishEnrich.sparql
9   xlink.categories.Fish.kbms.dbpedia_fish.templatequeries.enriching.parameter = [URI]
10  xlink.connect.radius = 1
11  xlink.fuzzy = true
12  xlink.fuzzy.value = 0.2
```

Configurability

- Configuration **API**
  - X-Link can be dynamically configured (even while a corresponding service is running)
  - Supported functions:
    - *Add a new category (using a resource class or a SPARQL query)*
    - *Update an existing category (using a resource class or a SPARQL query)*
    - *Remove a category*
    - *Change the displayed name of a category*
    - *Set/change the KBMs of a category*
    - *Set/change the resource classes, the SPARQL queries and the SPARQL template queries of a KBM*
    - *Set/change the active categories*
    - *Set/change the value of radius "r"*
    - *Set/change if fuzzy matching is allowed and the allowed edit distance percentage*

Portability

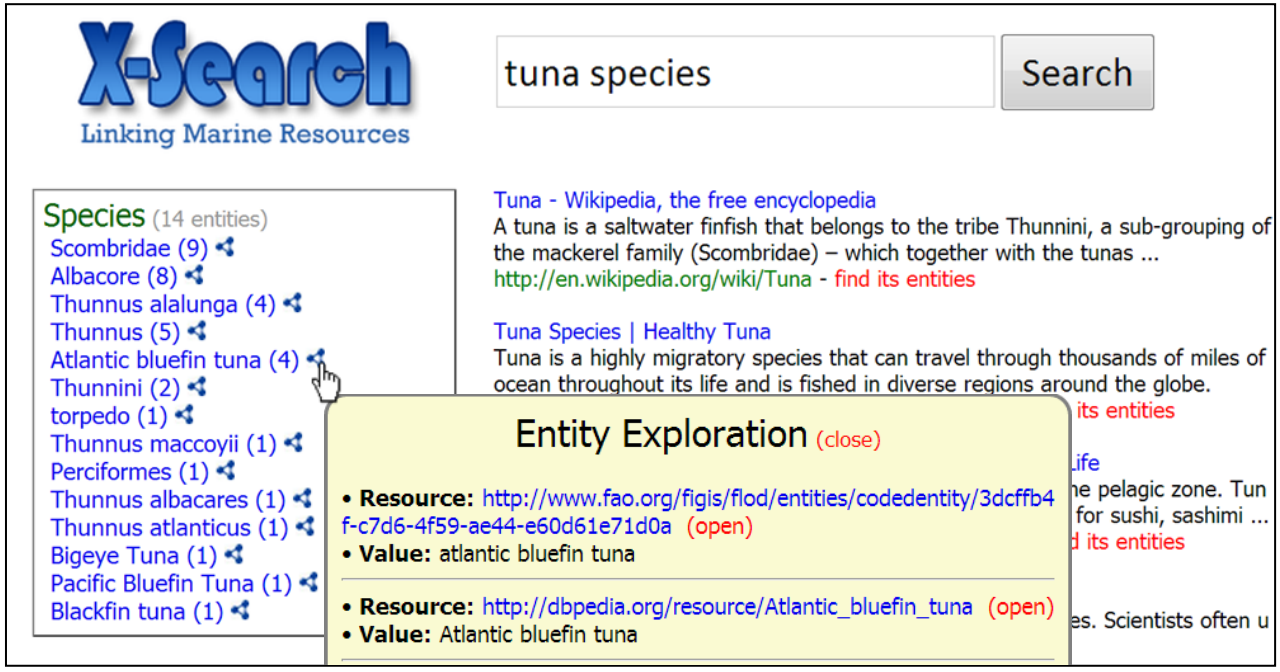
- The configuration files can be easily exchanged
  - Their size is relatively small
    - *E.g., for supporting 4 categories related to the marine domain, the configuration files have size less than 5MB*
  - The size mainly depends on the number of supported categories and on the number of the named entities in each category

- X-Link does not store any semantic information (e.g. URIs or RDF triples)
  - The entity linking and entity enrichment processes are performed at real-time

Applications (1/2)

http://62.217.127.118/x-search

- **X-Search** uses the X-Link library in two different contexts:
  - In the **marine** domain (in the context of the **iMarine** Project)
    - *The **MarineTLO**-based warehouse is exploited for entity linking and enrichment*
  - In **patent** search (in the context of the **PerFedPat** project)
    - *Tailored for medical biology*



Applications (2/2)

http://www.ics.forth.gr/isl/**Theophrastus**

- **Theophrastus**

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikimedia Shop

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

Print/export

Languages
العربية
Azərbaycanca
Català
Cebuano
Čeština
Dansk

Loricariidae

From Wikipedia, the free encyclopedia
(Redirected from Pleco)

The **Loricariidae** are the largest family of catfish (order Siluriformes), with 92 genera and just over 680 species to date, with new species being described each year.[3] Loricariids originate from freshwater habitats of Costa Rica, Panama, and tropical and subtropical South America. These fish are noted for the bony plates covering their bodies and their suckermouths. Several genera are sold as "plecos", notably the suckermouth catfish, *Hypostomus plecostomus*, and are popular as aquarium fish.

Common names [edit]

Hypostomus plecostomus, the original "plecostomus"

Members of t
commonly re
armoured ca
'plecos' or si
the species r
loricarids are
plecostomus.

These names are used practically interchangeably when referring to the Loricariidae. The name "Plecostomus" and

Loricariidae
Temporal range: Upper Miocene - Recent[1]

Superfamily: Loricarioidea
Family: **Loricariidae**
Rafinesque, 1815

Theophrastus - Entity Exploration (close)

- **URI:** http://dbpedia.org/resource/Hypostomus_plecostomus (open)
- **Name:** Hypostomus plecostomus
- **Binomial:** Hypostomus plecostomus
- **BinomialAuthority:** http://dbpedia.org/resource/Carl_Linnaeus (open)
- Show 'Same Genus'
- Show 'Same Species'
- Show 'Same Family'
- Show 'Same As'
- Show 'Taxonomy'
- Biodiversity Library
- ZooBank

(close)

Outline

- Introduction
  - *Named entity extraction / Motivation / Contribution*
- The proposed approach
  - *The configuration model*
- The system X-Link
  - *Architecture / Functionality / Configurability / Applications*
- Evaluation
  - *User study / Case study*
- Conclusion and Future Research

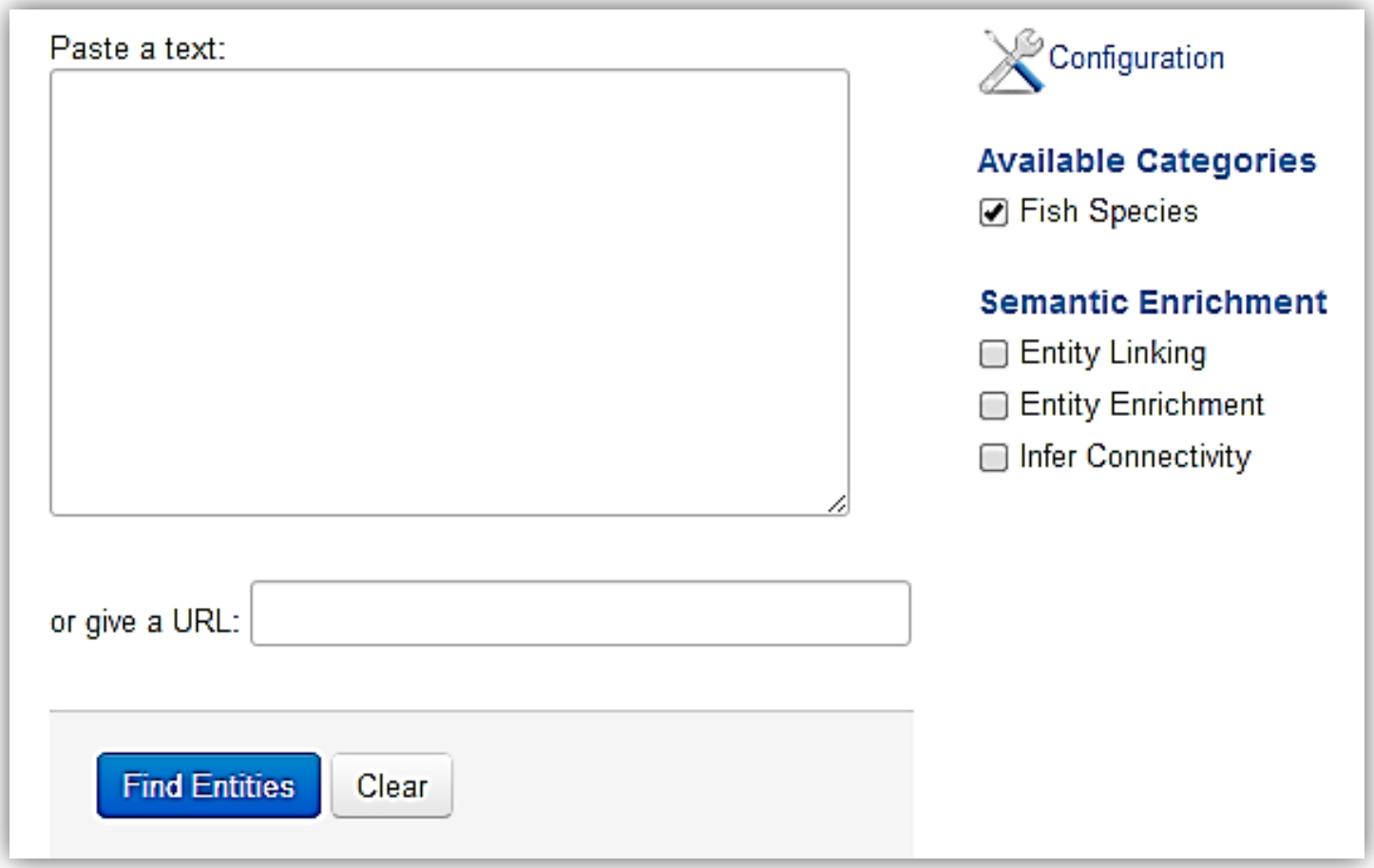
Task-based User Study

- Purpose:
  - Test the usability of the proposed approach
  - Identify usability problems
- X-Link was deployed as a Web application configured for the marine domain
  - Identification of Fish Species in a text or Web page
  - Entity Linking, Entity Enrichment
  - The administrator can change the configuration through an administration page

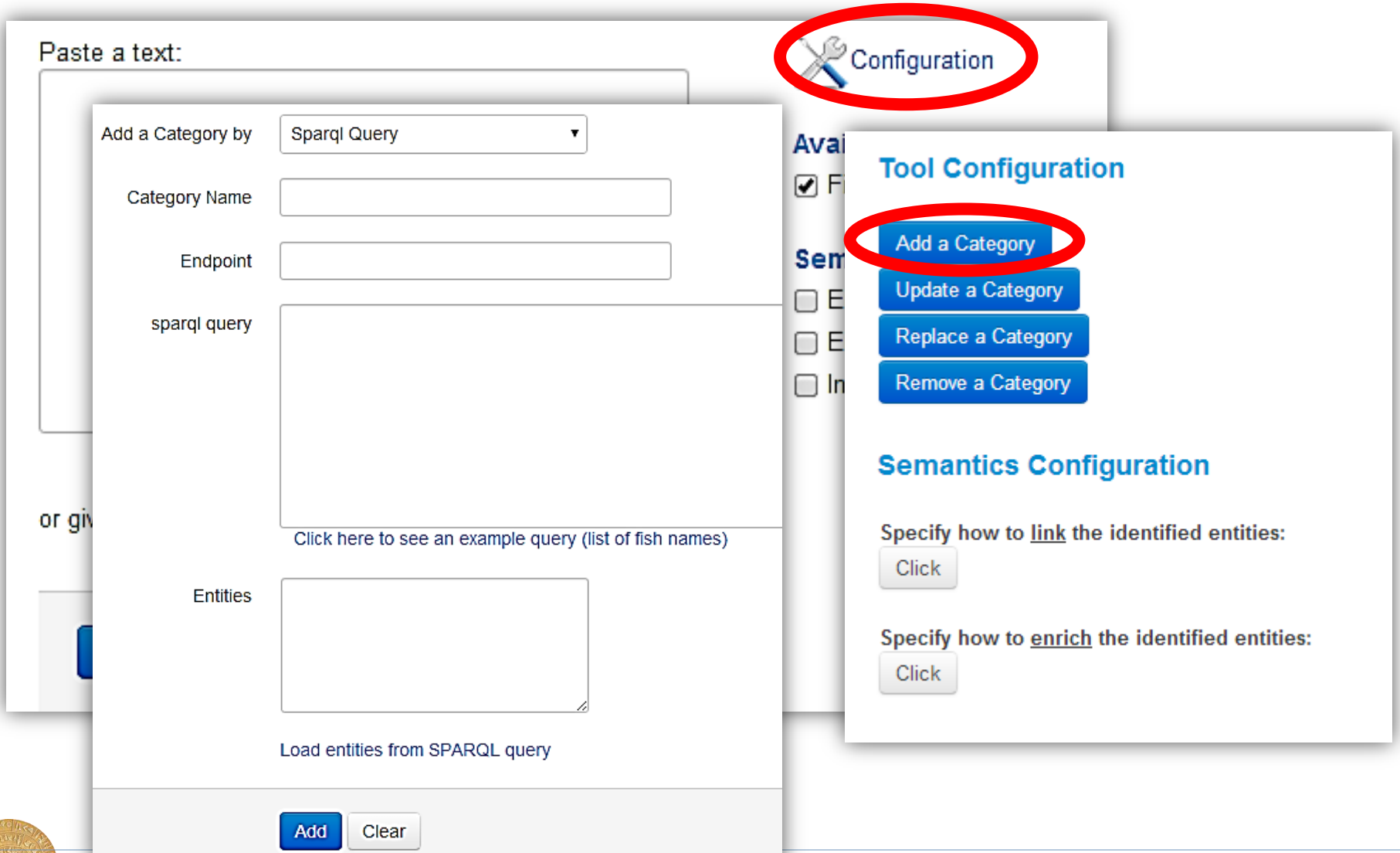
Target User:

An administrator or a developer who wants to use X-Link for building and dynamically configuring an application

Task-based User Study – The Web Application



Task-based User Study – The Web Application



Task-based User Study – Setting

- 11 subjects (23-34 years old) with computer science background, and basic knowledge of Linked Data and SPARQL
- 5-minute demonstration of the application and its functionality
- Tasks:
  - **(T1)** Add a new category of entities
  - **(T2)** Update a category
  - **(T3)** Specify how to link the identified entities of a category
  - **(T4)** Specify how to enrich the entity URIs of a category
  - **(T5)** Inspect the connectivity of the entity URIs (for r=1)
- The endpoint and the required RDF classes/properties are given
  - Since our objective was not to evaluate the ability of the user to find related semantic information
- We recorded:
  - Whether the subjects succeeded to complete each task
  - The time to successfully accomplish each task

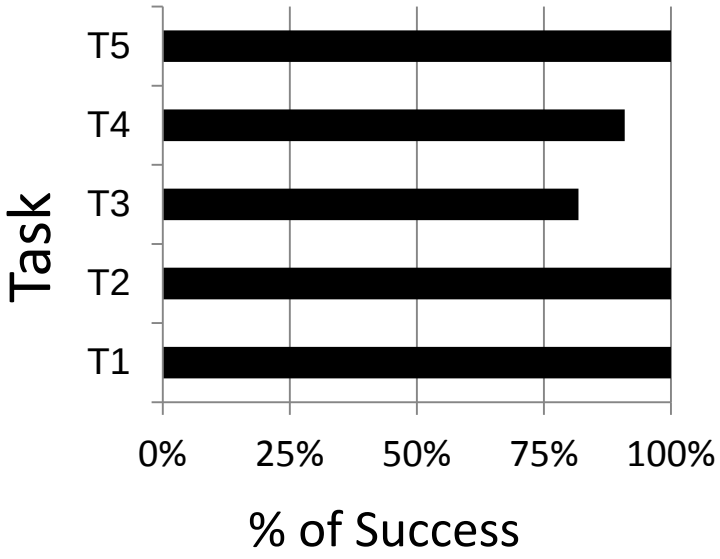
Task-based User Study – Scenario

Consider that you are the administrator of an application that can identify Fish names (currently supporting only the English language) in Web pages.

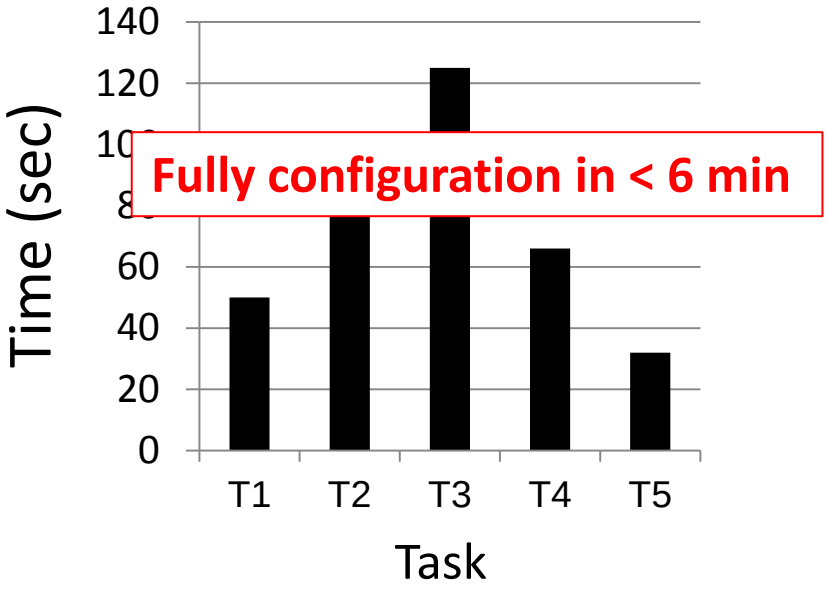
You have been asked to perform some changes. Specifically, by exploiting DBpedia, the application must also identify European Countries ***(T1)*** *as well as fish names in Spanish* ***(T2)*** *(because the application will be used mainly by Spaniards). Also, the identified fishes must be linked with resources from DBpedia* ***(T3)*** *and must be enriched with all their outgoing properties* ***(T4)****. Finally, in order to test that the system has been properly configured, perform entity mining in the Spanish version of Salmon's Wikipedia page and then inspect the connectivity of the identified entities* ***(T5)****.*

Task-based User Study – Results

Success rate for each task
(results from 11 users)



Average time for completing successfully each task



T1: Add the category "European Countries"
T2: Update the category Fish with fish names in Spanish
T3: Link the identified Fishes with resources from DBpedia
T4: Enrich the identified Fishes with their outgoing properties
T5: Inspect the connectivity of the identified entities

Task-based User Study – Questionnaire

(Q0) How easy was to configure the system according to the scenario?

(Q1) How easy was to add the new category of entities?

(Q2) How easy was to update the existing category?

(Q3) How easy was to specify how to link the identified entities?

(Q4) How easy was to specify how to enrich the identified entities?

(Q5) How easy was to inspect the connectivity of the identified entities?

(Q6) What was difficult for you during the execution of the scenario?

(Q7) How familiar are you with SPARQL?

Task-based User Study – Answers

(Q0-Q5) Evaluation of the difficulty in performing the scenario (results from 11 users)

Q	Very easy	Easy	Normal	Difficult	Very Difficult	Impossible
Q0	18%	82%	0%	0%	0%	0%
Q1	100%	0%	0%	0%	0%	0%
Q2	55%	27%	18%	0%	0%	0%
Q3	27%	45%	27%	0%	0%	0%
Q4	18%	55%	27%	0%	0%	0%
Q5	45%	45%	9%	0%	0%	0%

(Q6) What was difficult for you during the execution of the scenario:

- Difficulty in understanding the notion of the SPARLQ template queries
- Suggestion to provide a user-friendly interface for constructing them

(Q7) How familiar are you with SPARQL:

- 1 (I don't know SPARQL): 0%
- 2 18%
- 3 36%
- 4 36%
- 5 (I am expert in SPARQL) 9%

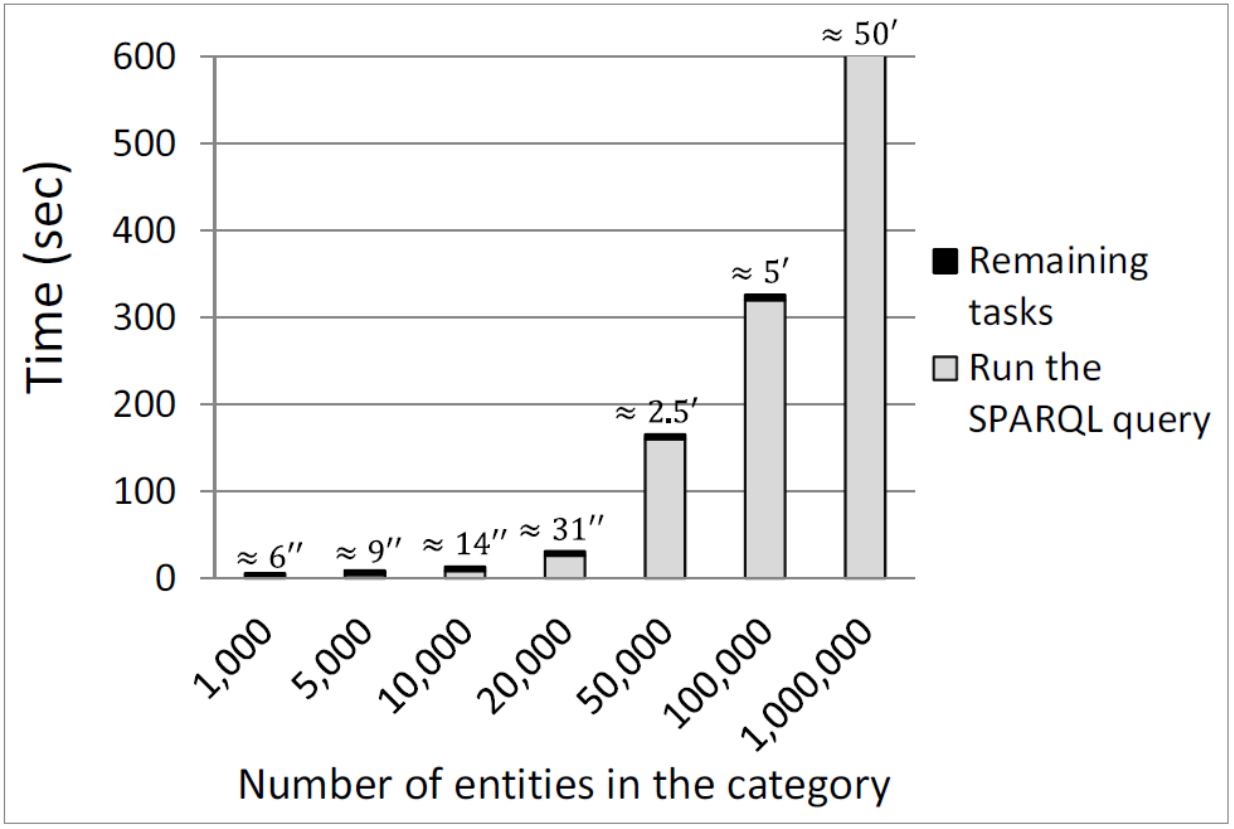
Case Study: Querying online DBpedia at real-time

- Purpose: Test the feasibility of the entire approach
- We measured the time for:
  - *creating a new category*
  - *linking an identified entity with semantic resources*
  - *enriching an entity URI*
  - *inferring the connectivity of the entity URIs*
- We repeated the experiments about 20 times and here we report the average values

- Data used in the experiments:
  - http://www.ics.forth.gr/isl/X-Link/files/exper_data.zip

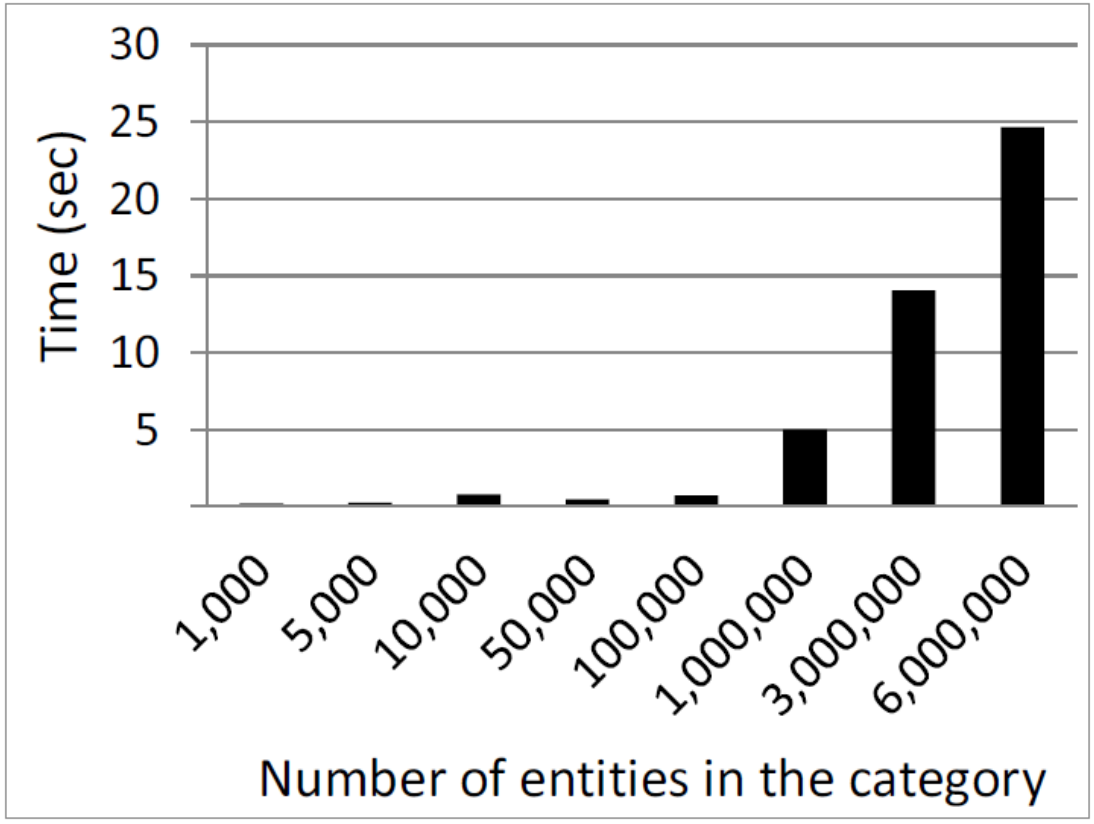
The experiments were carried out using an ordinary computer with processor Inter Core i7@3.4Ghz CPU, 8GB RAM, Win7 64bit. Implementation in Java 1.7.

Case Study: Time for adding a new category



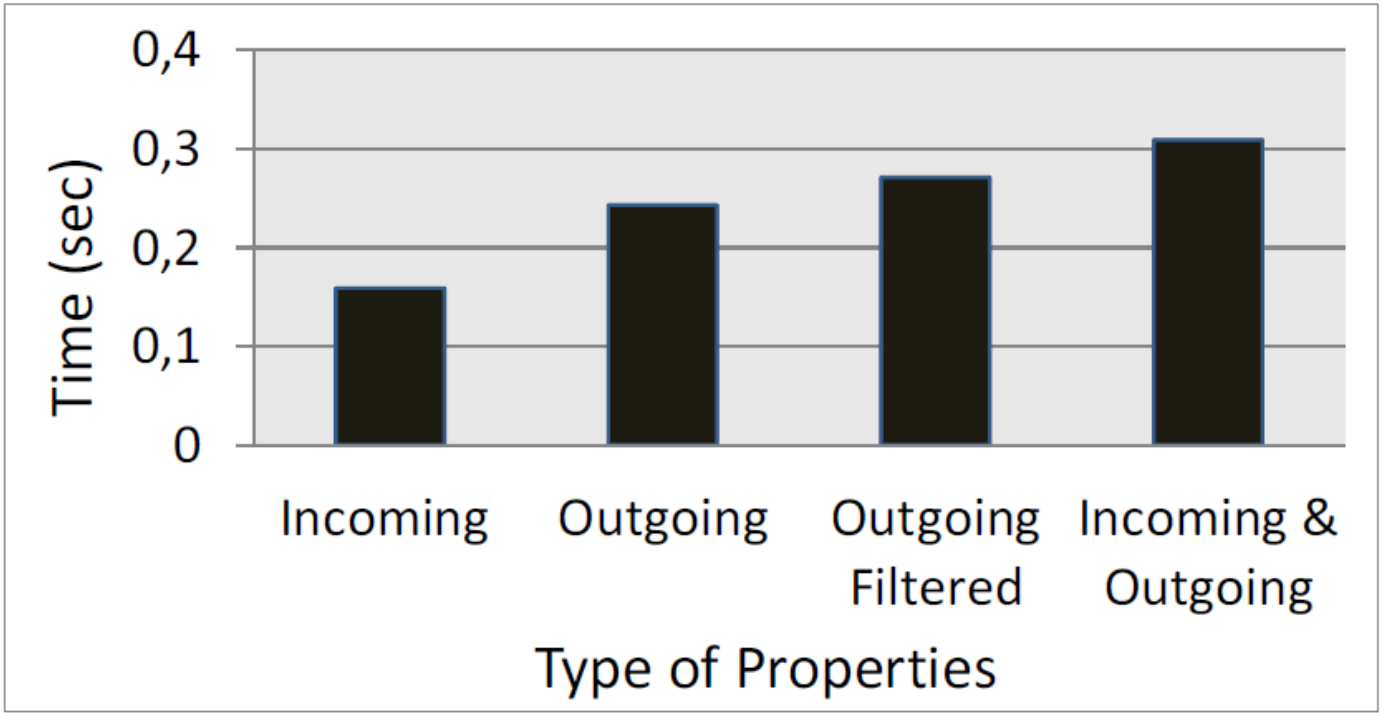
We used 7 sets of DBpedia resource classes.
Each set has 5 different resource classes containing a particular number of entities,
i.e. totally 35 different resource classes were used.

Case Study: Time for linking an identified entity



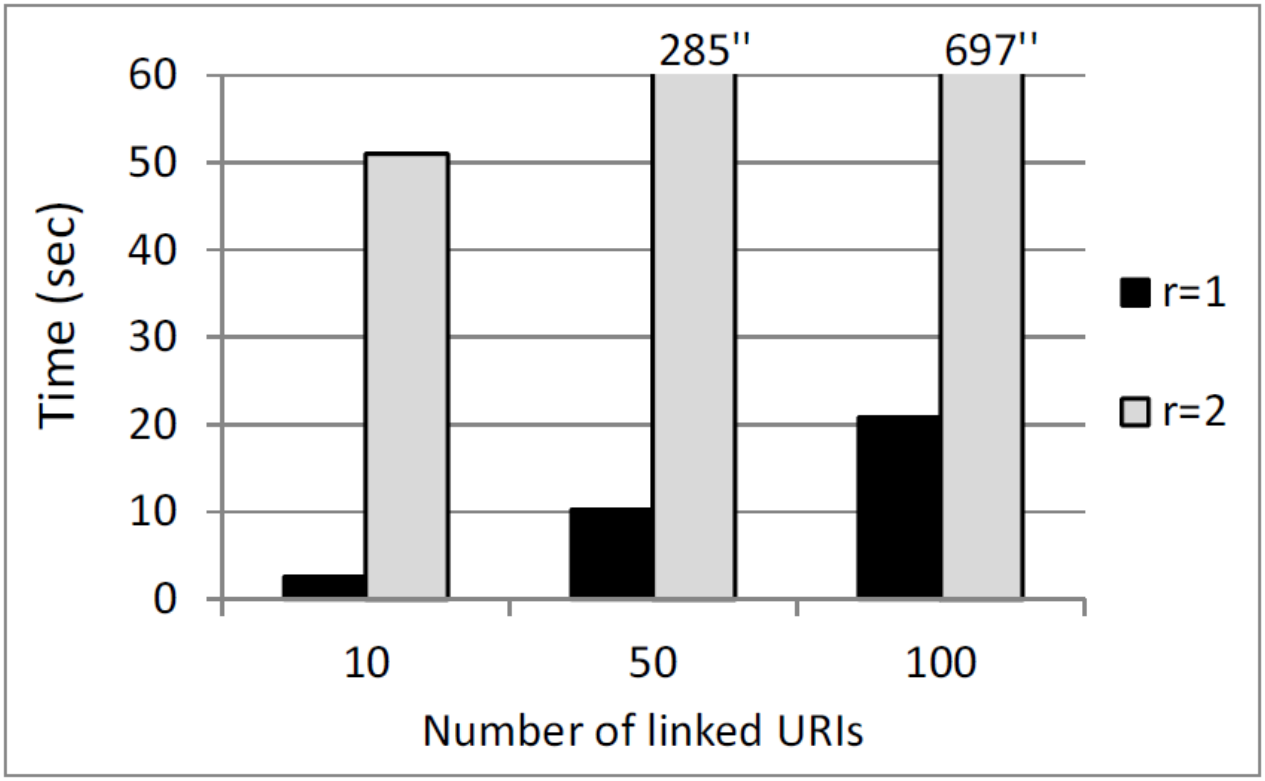
We used 8 sets of DBpedia resource classes.
Each set has 5 different resource classes containing a particular number of entities,
i.e. totally 40 different resource classes were used.
For each resource class, we randomly selected 10 labels of entities belonging to that class.

Case Study: Time for enriching an entity URI



We randomly selected 160 URIs from DBpedia

Case Study: Time for inspecting the connectivity



We randomly selected URIs of the same resource class from DBpedia and we repeated the experiments for 5 different resource classes.

Lessons Learned – Reliability and Scalability

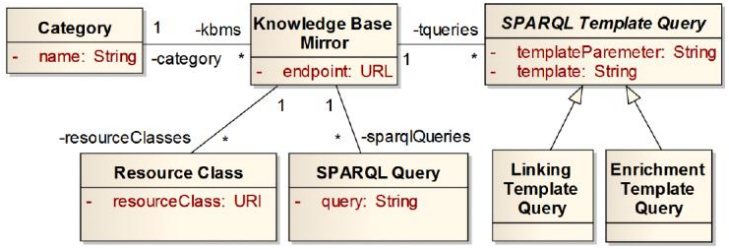
- Existing publicly available Knowledge Bases are not reliable
  - They mainly serve demonstration purposes
  - Their efficiency and availability change over time
  - They do not serve multiple concurrent requests
- If an entity belongs to a category with millions of entities then the linking time can be high
  - The same is true in case the underlying application requires to retrieve semantic information for numerous entities at once
  - Caching/Indexing is a solution, but with the cost of loosing the freshness of the results
- In a real application:
  - The underlying KBs may not be publicly available
  - A dedicated **Warehouse** can be constructed that will serve the application
  - Distributed infrastructure

Outline

- Introduction
  - *Named entity extraction / Motivation / Contribution*
- The proposed approach
  - *The configuration model*
- The system X-Link
  - *Architecture / Functionality / Configurability / Applications*
- Evaluation
  - *User study / Case study*
- Conclusion and Future Research

Conclusion

- A generic model for configuring (dynamically) a LOD-based NEE system



- X-Link
  - A LOD-based, fully configurable, NEE tool that supports the proposed model
- By adopting the proposed approach one can configure a NEE system within a few minutes
- The exploitation of LOD can be supported at query-time
- The major bottleneck is the reliability and performance of online SPARQL endpoints
  - *We expect this limitation to get overcome in the near future*
  - *In the meanwhile, we can use caching/indexing/dedicated warehouses/distributed infrastructure*

Future Research

- **It would be beneficial for the community if every NEE system supported the proposed configuration model**
  - » We work on defining an RDF vocabulary with explicit semantics

- We evaluate approaches for entity disambiguation that are appropriate in our setting

Argentina, the country

Argentina, the fish genus

- We elaborate on methods for ranking the matching URIs in case they are numerous

Thank you!

http://www.ics.forth.gr/isl/X-Link/