Introduction

Something about me

- 2005 Studies of Computer Science, University Halle (Germany)
- 2010 PhD in Software Engineering and Programming Languages, University Halle (Germany)
- 2012 Project Lead of “Semantic Web Project” (R & D), Unister GmbH (Germany)
- 2015 Head of Research and Development Department, Unister GmbH (Germany)
- 2016 Research and Development Lead Mercateo AG (Germany)
- 11/2016 - Head of Architecture, Web Technology and IT Research, DATEV eG (Germany)

DATEV eG

- Software company and IT service provider
- turnover: > 900 million euros
- age: > 50 years old
- core market: Germany
- fields: accounting, business consulting, taxation, enterprise resource planning (ERP) as well as organization and planning
- members: > 40.000
- customers: > 2.6 million companies


Qanary

Qanary

Several tasks common across QA systems.

Having these tasks as components, sharing a vocabulary can help develop better QA system.

Qanary

A framework for developing QA system by integrating various component using a pre-defined **Q**uestion **An**swering vocabul**ary**.

Observations

- Limited compatibility
- Predefined pipelines
- Limited semantics

+ Interoperable infrastructure
+ Exchangeable components
+ Flexible granularity
+ Isolation of components

Goals

Easy-to-build QA systems on-top of reusable components

Establish an ecosystem of components for QA systems

Qanary Methodology and Technical Framework

Knowledge perspective - I

Requirements of Knowledge perspective

- ❏ abstract knowledge representation: qa vocabulary
- ❏ align the input/output of the each component in a QA process

QA vocabulary

Represent all the available knowledge about a question.

- ❏ representation of knowledge about question separated from process
- ❏ includes trust & provenance
- ❏ self-describing, reusable and extensible
- ❏ enables efficient collaboration on a data-level
- ❏ agnostic to question format (text, structured, audio)
- ❏ agnostic to question answering processing steps and implementation

Aligning I/O in QA processes

- ❏ required input mapped from KB
- ❏ computed output mapped into KB
- ❏ mapping on a logical and sound level

Knowledge Representation using the qa Vocabulary

Abstract Knowledge Representation (KR)

Represent all the knowledge about a question using a RDF vocabulary

KR requirements

- ❏ self-describing, sound knowledge representation
- ❏ represent provenance for (all) information
- ❏ represent trust for (all) information

Derived Technology stack

- Resource Description Framework (RDF)
- Web Annotation Data Model (WADM)
- Question Answering vocabulary (qa)

Resource Description Framework (RDF)

Introduction to RDF (slides by Manolis Koubarakis)

Web Annotation Data Model (WADM)

- oa:Annotation
- oa:hasTarget
- oa:hasBody
- oa:annotatedAt
- oa:annotatedBy

```
<myIRI> a oa:Annotation;
      oa:hasTarget <questionIRI> ;
      oa:hasBody <TextSelector> ;
      oa:annotatedBy <DBpediaSpotlight> ;
oa:annotatedAt "..."^^xsd:date ;
```

QA Vocabulary

Introducing new QA-related concepts on-top of WADM

```
qa:Question
     rdfs:subClassOf oa:Annotation.
qa:Answer, . . .
qa:Dataset, . . .
qa:AnnotationQuestion, . . .
```

From KR to methodology

Conclusion: Advantages of using an ontology

- agnostic to question format (text, structured, audio)
- agnostic to question answering processing steps
- agnostic to implementation
  - programming language
  - component granularity

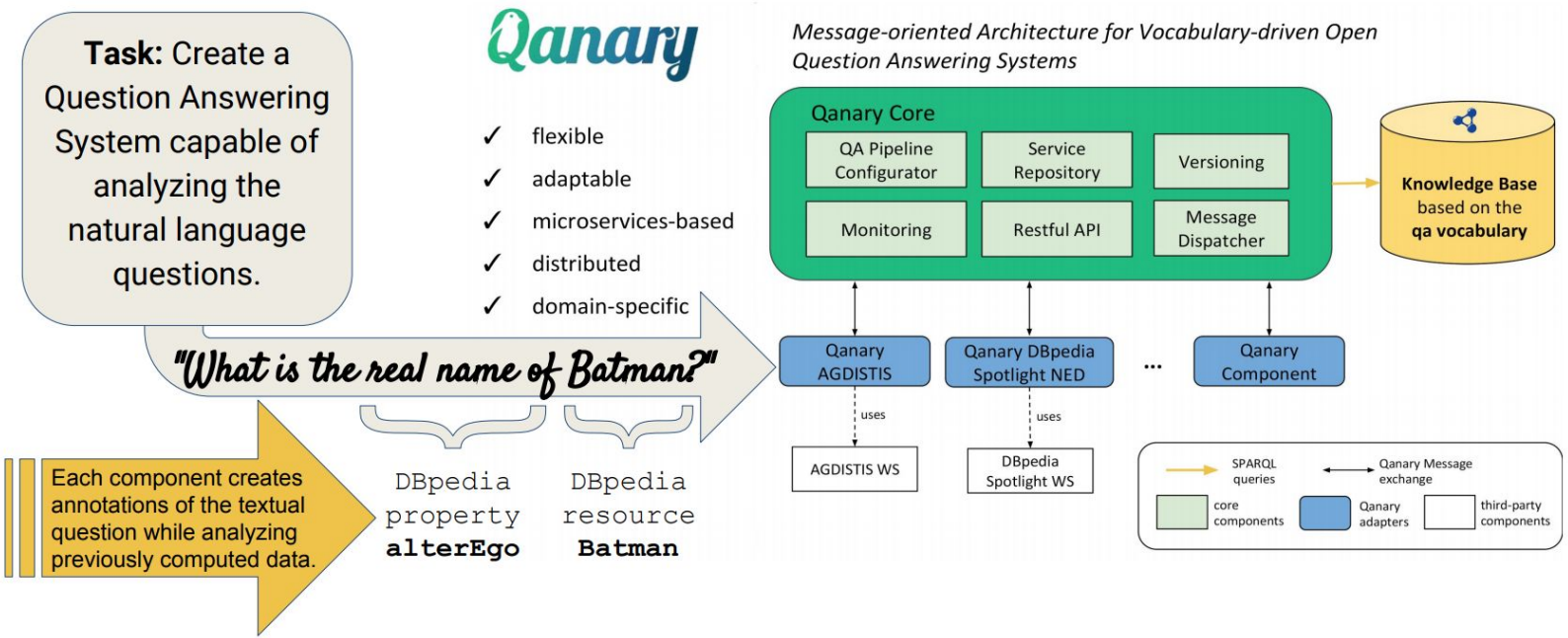
From KR to methodology II

Methodology:

- abstract knowledge representation
  - independent representation
- align the input/output of the each component
  - on a logical and sound level

Methodology

Qanary Architecture



It’s about the components, stupid

An agile QA framework can only provide common features

- `central data access, logging`

Any particular problem solving/algorithm needs to be separated from the pipeline

It’s about the components, stupid -II

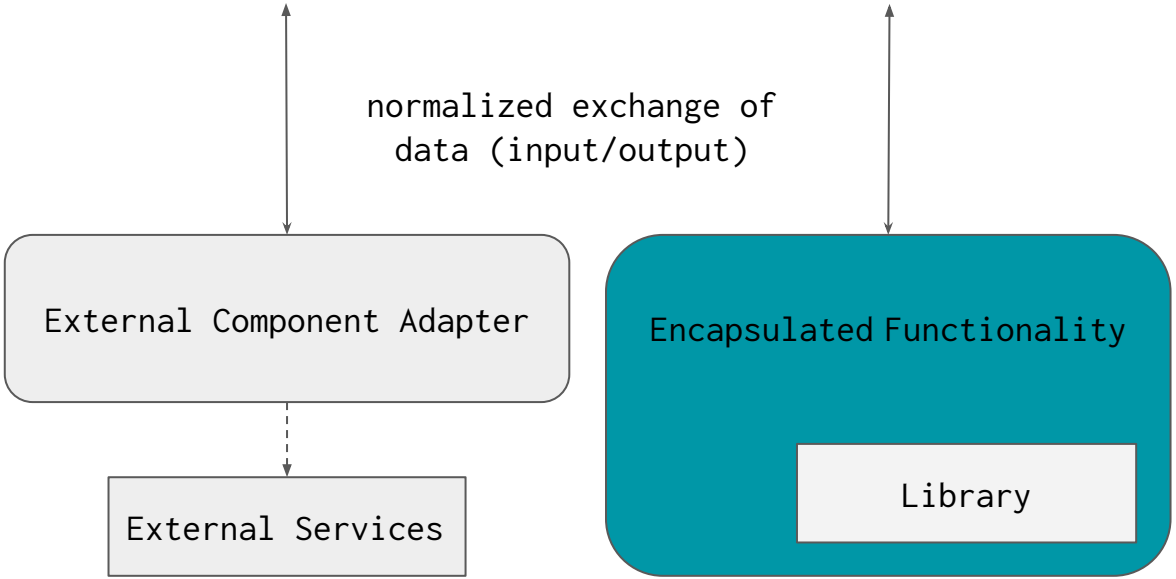
create exchangeable, isolated components only communicating via data.

component data needs to be mapped/aligned to the data of the QA process.

Component Data Alignment- CDA

Goal: Establish common ground for the research community

CDA- II


normalized exchange of
data (input/output)
External Component Adapter
External Services
Encapsulated Functionality
Library

CDA III

Alignment of input/output of each component with qa, if component provides output using presentation as

- semantic data (RDF)
  - logical representation of alignment
    - ontology alignment (OWL,DOL)
    - SPARQL query
  - non-semantic data (API, JSON, XML, CSV)
    - SPARQL query

- NER/NED
  - DBpedia Spotlight [Mendes et al., 2011] (NIF)
- relation detection
  - PATTY [Nakashole et al., 2012]
- query construction
  - SINA [Shekarpouret al., 2011]

CDA: NED

Create component input

- fetch question URI (from Qanary triplestore)

Processing

- retrieve textual question representation from URI
- compute named entities within the text

store component output

- for each named entity:
  - create a oa:TextSelector within the Qanary triplestore containing the positions of the particular Named Entity

CDA: NED Benefits

Easily replace the NED component.

Measure quality against exchangeable relation detection and query construction components.

CDA: Relation Detection

Create component input

- fetch question URI (from Qanary triplestore)
- fetch Named Entities which are already available

Processing

- retrieve textual question representation from URI
- compute relations within the text

store component output

- for each named relation:
  - create a relation resource within the Qanary triplestore (using a oa:TextSelector to mark the positions)

CDA: Relation Detection Benefits

Any improvement on the NED component (i.e., replace) will improve the quality here

Measure quality against exchangeable query construction components.

CDA: Query Construction

Create component input

- fetch Named Entities (which are already available)
- fetch Relations (which are already available)

Processing

- compute SPARQL

Store component output

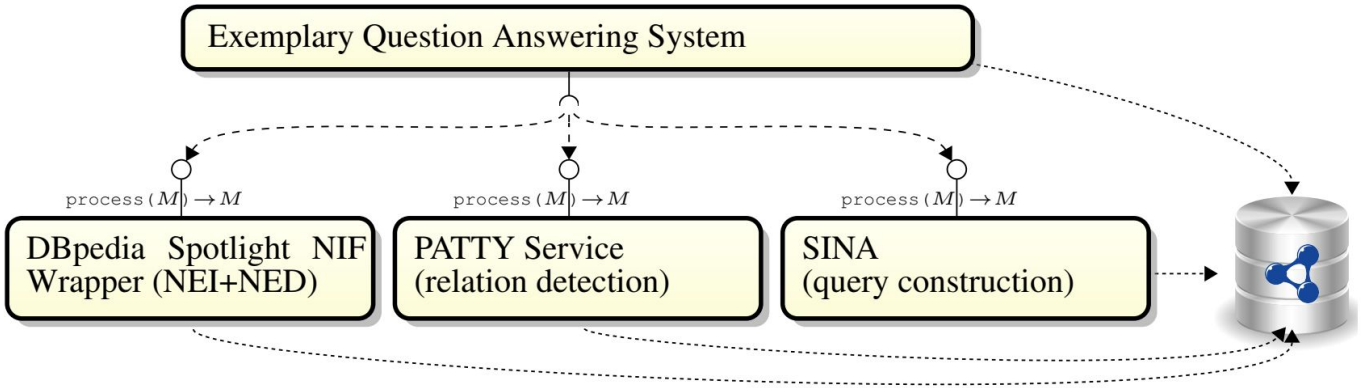
- for each created SPARQL:
  - store a resource/SPARQL in the Qanary knowledge base

CDA: Query Construction Benefits

Any improvement on the NED component (i.e., replace) will improve the quality here

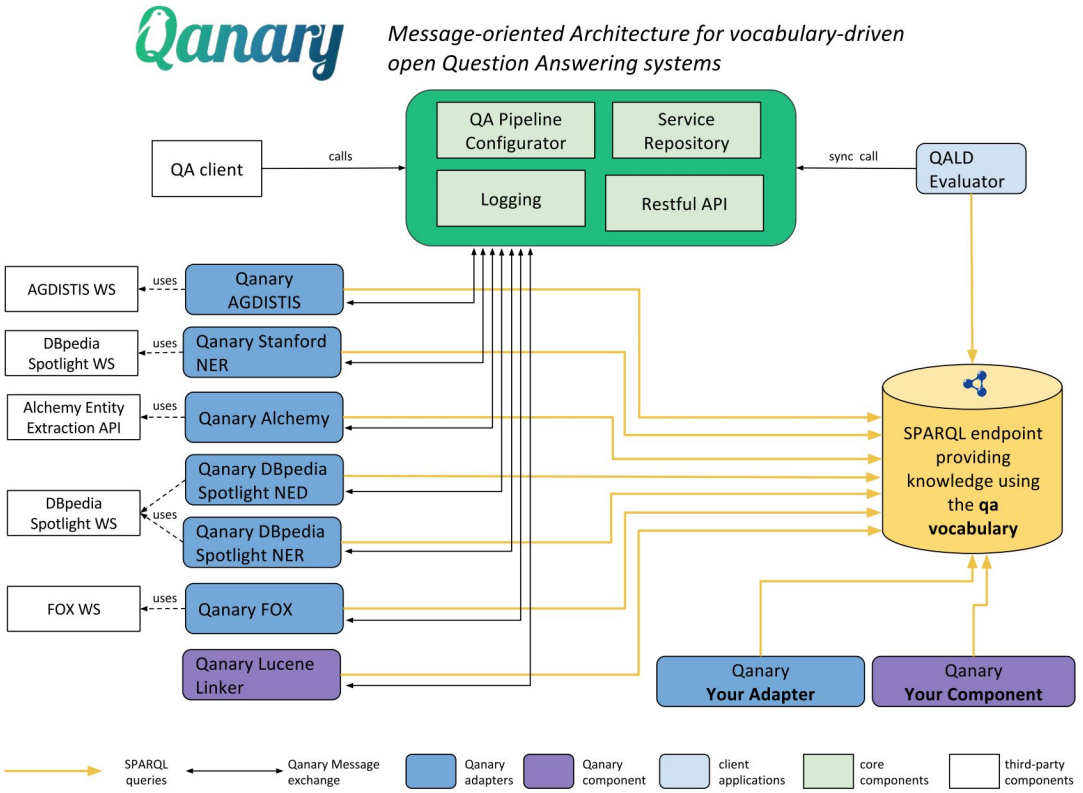
Any improvement on the Relation detection component (i.e., replace) will improve the quality here

Case Study



Component		Process within components
DBpedia Spotlight		Retrieve data from KB
PATTY		Process data
SINA query execution		Extend KB

CDA- II


Qanary
Message-oriented Architecture for vocabulary-driven open Question Answering systems
QA client
calls
QA Pipeline Configurator
Service Repository
Logging
Restful API
sync call
QALD Evaluator
AGDISTIS WS
uses
Qanary AGDISTIS
DBpedia Spotlight WS
uses
Qanary Stanford NER
Alchemy Entity Extraction API
uses
Qanary Alchemy
DBpedia Spotlight WS
uses
Qanary DBpedia Spotlight NED
Qanary DBpedia Spotlight NER
FOX WS
uses
Qanary FOX
Qanary Lucene Linker
SPARQL endpoint providing knowledge using the qa vocabulary
Qanary Your Adapter
Qanary Your Component
SPARQL queries
Qanary Message exchange
Qanary adapters
Qanary component
client applications
core components
third-party components

Coffee Break

In the following practical session you will need:

- Internet connection
- text editor or any Ontology Designer
- Git client
- Java and Maven
- Stardog triplestore (free version)

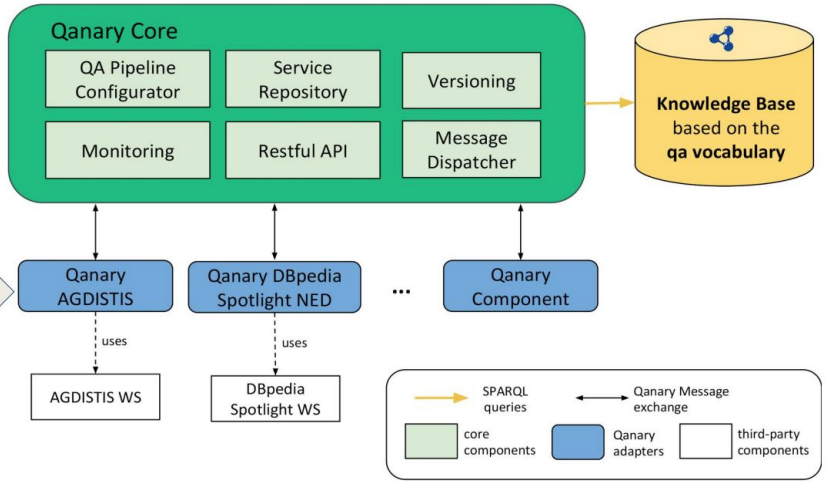
Our Goal

Task: Create a Question Answering System capable of analyzing the natural language questions.

Qanary

- ✓ flexible
- ✓ adaptable
- ✓ microservices-based
- ✓ distributed
- ✓ domain-specific

Message-oriented Architecture for Vocabulary-driven Open Question Answering Systems



"What is the real name of Batman?"

Each component creates annotations of the textual question while analyzing previously computed data.

DBpedia property **alterEgo**

DBpedia resource **Batman**

Preparation

Given questions:

- “Name all the movies in which Robert Downey Jr acted?”
- “Is Bruce Wayne the real name of Batman?”
- “How many partners had Batman?”

Preparation - Task

Model the required annotations for answering these questions

Write the SPARQL query to retrieve the answers for these queries

Note: Typically, the result of a QA process is not a SPARQL query. Due to time constraints, we exclude the mostly following Answer Generation (e.g., using Natural Language Generation or visualizations) from this exercise. See wolframalpha.com from inspiration

Coding Session: Implement your first QA system from existing components

Implement your first QA system

https://github.com/WDAqua/Qanary/wiki/Demo:-How-to-Create-a-Question-Answering-System-capable-of-Analyzing-the-Question-%22What-is-the-real-name-of-Batman%3F%22

Implement your first QA system

- ❑ Git checkout Qanary ecosystem’s components
- ❑ Run components
- ❑ Run Qanary QA system template
- ❑ Configure your pipeline
- ❑ Run the pipeline
- ❑ Test your QA system with some questions on DBpedia
- ❑ Done

Validate the quality of your QA system

Implement your first QA system

Interactive validation using TRILL front-end from Qanary Ecosystem

Use Qanary QALD validator to compute precision, recall and f-measure

Improve and revalidate your QA system

Improve and revalidate your QA system

Solve questions not implemented before

- ❑ Pick from prepared list
- ❑ Define test cases
- ❑ Extend functionality
- ❑ Validate results in triplestore

Solve new QA tasks

Extend the qa vocabulary

Choose existing QA components supporting your task

Implement new QA component for your new use case

Extend test cases and validate your work

Final Remarks

Summary

Qanary methodology, the RDF vocabulary qa and the corresponding component-oriented Qanary framework.

Advantage of the Qanary ecosystem for rapid research results

Learn to iteratively build, validate and improve your own QA system using the Qanary framework

You built a QA system capable of answering generic question in a specific domain

Take Away: Qanary methodology

Qanary: knowledge-driven methodology for QA systems

Build on-top of the qa vocabulary (i.e., knowledge-driven approach)

Agile approach for creating QA systems

Join Qanary at GitHub

github.com/WDAqua/Qanary

References.

Mendes, Pablo N., et al. "**DBpedia spotlight: shedding light on the web of documents.**" Proceedings of the 7th international conference on semantic systems. ACM, 2011.

Shekarpour, Saeedeh. "**Semantic Interpretation Of User Queries For Question Answering On Interlinked Data.**" (2015).

Nakashole, Ndapandula, Gerhard Weikum, and Fabian Suchanek. "**PATTY: a taxonomy of relational patterns with semantic types.**" Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, 2012.