Welcome to INFO216: Knowledge Graphs

Spring 2023

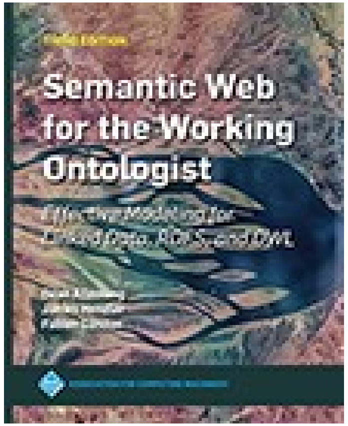
Andreas L Opdahl
<Andreas.Opdahl@uib.no>

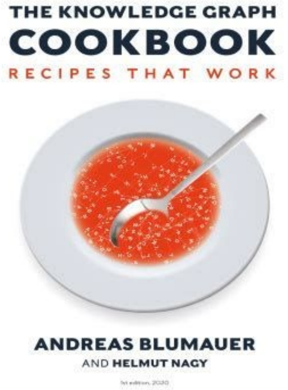
Session 12: Enterprise Knowledge Graphs II

- Themes:
  - Open OGs *(← S04-S05)*
    - Linked Open Data resources / datasets
    - Wikidata, DBpedia, GDELT, EventKG GeoNames, WordNet, BabelNet...
  - Enterprise KGs I (→ *S06)*
  - Enterprise KGs II:
    - Google's knowledge graph
    - Amazon's product graphs
    - the News Hunter infrastructure and architecture
  - *JSON-LD*

Readings

- Sources (suggested):
  - Blumauer & Nagy (2020):
    Knowledge Graph Cookbook – Recipes that Work:
    parts 2 and 4
- Resources in the wiki <http://wiki.uib.no/info216>:
  - *Introducing the Knowledge Graph: Things not Strings*, Amit Singhal, Google (2012)
  - *A reintroduction to our Knowledge Graph and knowledge panels*, Danny Sullivan, Google (2020)
  - *How Amazon's Product Graph is helping customers find products more easily*, Arun Krishnan, Amazon (2018)





Is anyone really using Knowledge Graphs?

Is anyone really using this?

Yes!

Tencent 腾讯

UniProt

USGS

Google

Bing

Alibaba.com

Baidu 百度

PubMed

facebook

ANTONI VAN LEEUWENHOEK FOUNDATION

MAASTRO

DEUTSCHE NATIONAL BIBLIOTHEK

The New York Times

BBC

europeana

NXP

REUTERS

KUNGL. BIBLIOTEKET

National Library of Sweden

RENAULT

EPA United States Environmental Protection Agency

IOS Press

European Commission

Walmart

SIEMENS

BEST BUY

Deloitte.

SPRINGER NATURE

amazon.com

accenture

ELSEVIER

Is anyone really using this?

Yes!

- But...
  - not quite as in the semantic web vision
  - not quite as in the LOD vision either
- Knowledge graphs are (additionally) becoming:
  - company internal
  - based on other technologies
    - such as general graph databases
  - not always linked to the LOD cloud

Is anyone really using this?

Yes!

- But...
  - not quite as in the semantic web vision
  - not quite as in the LOD vision either
- Knowledge graphs are (additionally) becoming:
  - company internal
  - based on other technologies
    - such as general graph databases
  - not always linked to the LOD cloud

Many of these ideas
are widely adopted too, such as:

- microdata / schema.org
- RDF / SPARQL / … for semantic data exchange
- graph representations in general

Is anyone really using this?

Yes!

- But...
  - not quite as in the semantic web vision
  - not quite as in the LOD vision either
- Knowledge graphs are (additionally) becoming:
  - company internal
  - based on other technologies
    - such as general graph databases
  - not always linked to the LOD cloud

Similar ideas,
adapted to new uses
and business contexts,
using a combination of
standard and other
technologies

Google's
Knowledge Graph

Google's Knowledge Graph

- Google Knowledge Graph (from 2012)
  - "Things, not Strings"
  - seeded from Freebase
  - facts from Wikipedia, Wikidata, CIA World Factbook
    - a growing number of other sources
  - enriched by natural-language parsing (NLP)
    - Google's Knowledge Vault
  - used internally for many purposes
  - visible in Google Search results (Knowledge Panels)
  - question answering in Google Assistant / Home

Caution: *The public documentation is limited, so this is compiled based on presentations, technical notes, forums etc.*

Google's Knowledge Graph

- Coverage:
  - claimed
    - 18 billion facts (18G, norsk: 18 milliarder) about 570 million entities *soon after start*
  - 70 billion facts claimed in (2016)
  - 500 billion facts about five billion entities (2020)
    - ...perhaps 3 times the size of the LOD cloud
  - from English to multiple languages
- Critiques:
  - source attribution, incl. Wikipedia / Wikidata

Caution: *The public documentation is limited, so this is compiled based on presentations, technical notes, forums etc.*

Google's Knowledge Vault Project

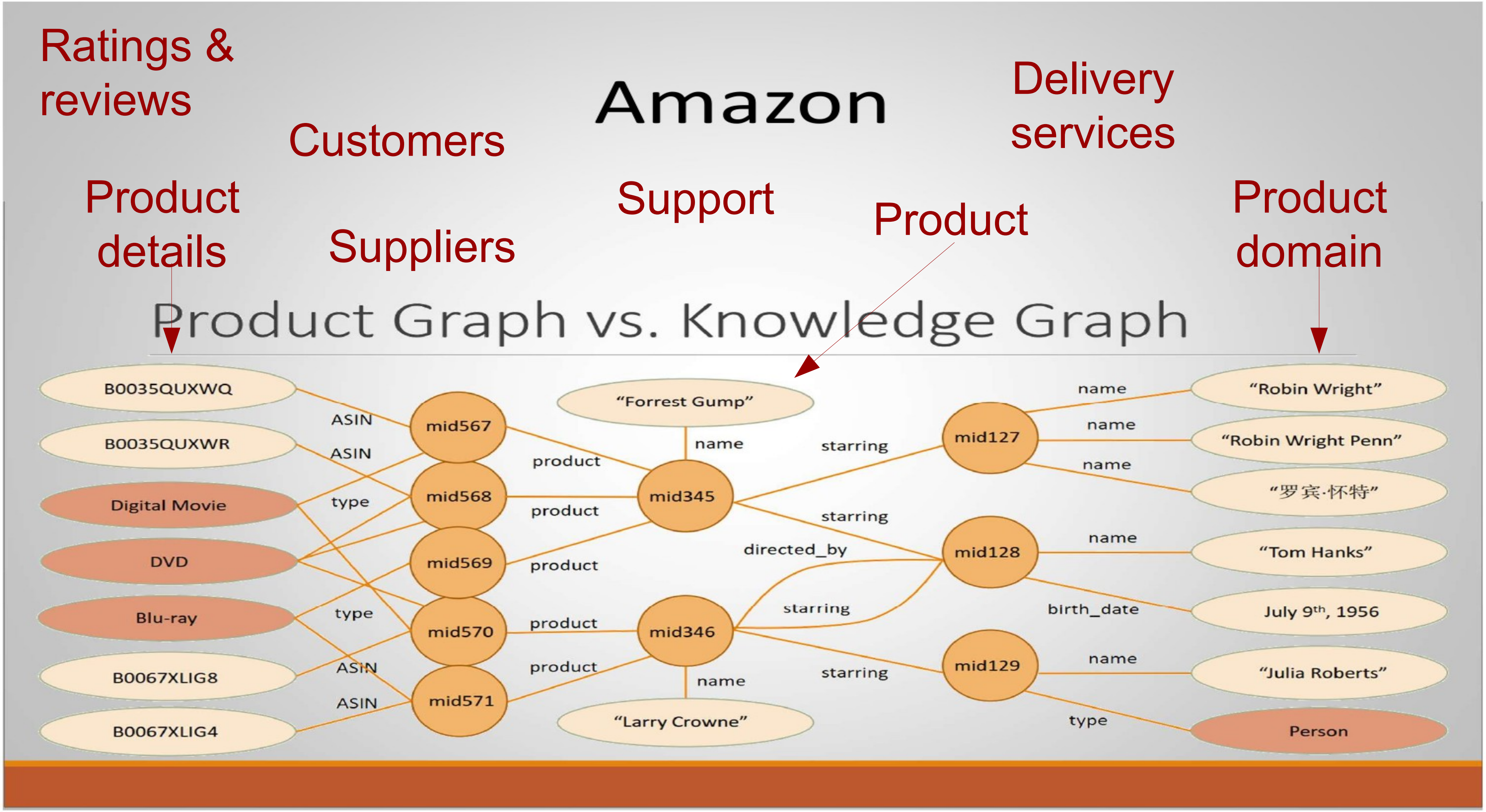
- Google Knowledge Vault
  - extends the Knowledge Graph
  - covers resources not from open semantic datasets
  - facts extracted from the whole web
    - NLP of text documents
    - HTML trees and tables
    - human annotated pages (e.g., schema.org)
  - probabilistic reasoning
    - graph-based priors
    - knowledge fusion

***Caution:** The public documentation is limited, so this is compiled based on presentations, technical notes, forums etc.*

Amazon's Knowledge Graph

Amazon's ambition (←S01)

- Let shoppers find the best products that fit their needs
  - allow greater variation in search terms
  - allow complex queries
- Ambition: *to structure all of the world's information as it relates to everything available on Amazon*
- Describe every product on Amazon
  - both products and non-products
  - both concrete and abstract concepts
  - link related entities, both internal and external
- Enhanced customer experience
  - visit Amazon to see what's new or interesting
  - discover ways to simplify and enrich their lives



Frank van Harmelen (2018): Keynote at CAiSE’18

Challenges

- Ingest product-related information from Amazon's detail pages and from the Internet at large
  - product information is largely unstructured
  - trustworthiness of sources
- Machine learning techniques for
  - knowledge extraction, linkage and cleaning
  - distantly supervised learning
    - train on more structured subset of data
    - run on larger unstructured data space
  - open information extraction
  - graph mining techniques to identify interesting hidden patterns (buying product-X → buying product-Y)

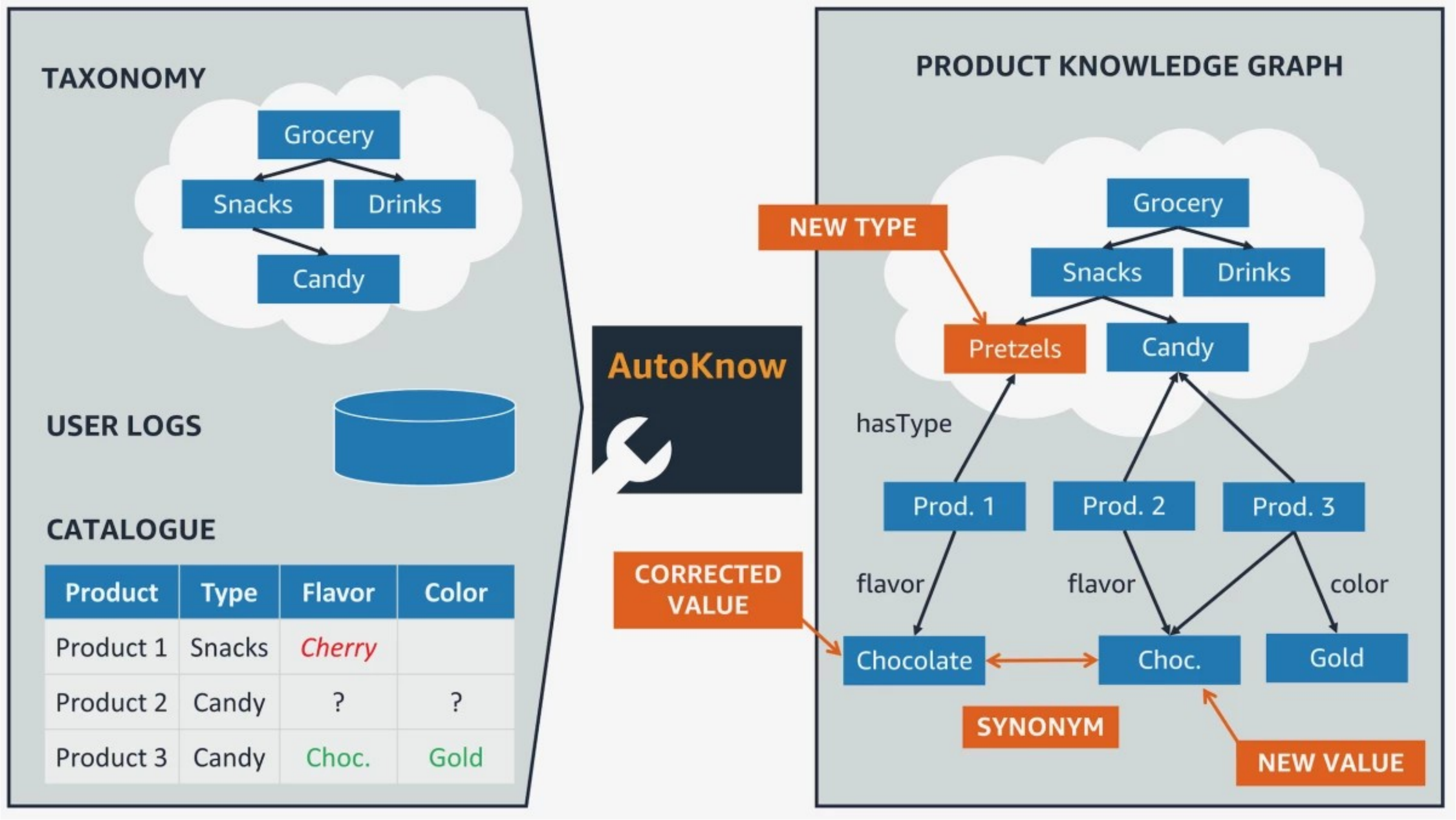
Challenges

- *There is an enormous amount of data* - just for products...
- Can the KG be built/grown automatically?
- Automatic KG building:
  - most successful in stable domains (e.g., movie information)
    - few types and properties
    - many available sources of structured data
  - retail data is an evolving and unbounded domain
    - many product types
    - even more product properties
      (color for clothes, flavor for candy, wattage for electronics, ...)
    - new types and properties all the time
    - much of the information is unstructured / NL text:
    - product descriptions (“the coffee mug get too hot to hold”), customer reviews, questions-and-answers (QA) forums

Amazon’s AutoKnow

- A project and software platform to automatically grow and maintain Amazon’s Product Graph (the product information)
  - heavily based on ML techniques
- Inputs:
  - existing (and growing) product taxonomy (aka rdfs:subClassOf tree)
    - which also a product graph, because it includes product properties
  - a catalogue of products that includes:
    - structured information: labelled product names, product sheets, ...
    - unstructured product descriptions as free text
  - free-form product-related information:
    customer reviews, product-related QAs, product query data, ...

Amazon's AutoKnow



Tasks:

- adding new product types
- adding new product values
- correcting product values
- identifying synonyms

Amazon’s AutoKnow

- Five modules (ontology suite + data suite):
  1) *taxonomy enrichment* extends the number of entity types in the graph
  2) *relation discovery* identifies attributes of products, those attributes’ range of possible values (different flavors or colors, for instance), and, crucially, which of those attributes are important to customers
  3) *data imputation* uses the entity types and relations discovered by the previous modules to determine whether free-form text associated with products contains any information missing from the graph
  4) *data cleaning* sorts through existing and newly extracted data to see whether any of it was misclassified in the source texts
  5) *synonym finding* attempts to identify entity types and attribute values that have the same meaning

Taxonomy enrichment module

- Extends the number of entity types in the graph
- Labelling of product titles in the source catalogues:
  - labelling of product types:
    "Ben & Jerry's black cherry cheesecake <u>ice cream</u>"
  - labelling of product property values:
    "Ben & Jerry's <u>black cherry cheesecake</u> ice cream" (flavour)
    "<u>Ben & Jerry's</u> black cherry cheesecake ice cream" (brand)
  - ML model trained on Amazon's existing hand-made product data
- Hypernym classification (aka rdfs:subClassOf):
  - "Ice cream" → "Ice cream and novelties" → "Frozen"
  - data from customer interactions:
    which products customers viewed or purchased after a single query
  - ML model trained on manually-labelled product data

Relation discovery module

- Identifies
  - properties (attributes) of products
  - ranges of property values (e.g., different flavours or colors)
  - *which of those attributes are important to customers*
- Two classifiers:
  - whether the property applies to a given product
    (flavor applies to food but not to clothes)
  - how important the attribute is to buyers
    (brand name is more important for snack foods than for produce)
  - input data:
    - from providers (product descriptions)
    - from customers (reviews and Q&As)
  - trained on manually annotated data

Data imputation module

- Determine whether free-form text associated with products contains any information missing from the graph
- Looks for terms in product descriptions that
  - may fit the new product and attribute categories
  - but that are not yet in the graph.
- Term (word) embeddings ($\leftarrow$S11)
  - represents descriptive terms as points in a vector space
  - related terms are grouped together
  - *if a number of terms clustered together in the space share the same property or product type, the unlabelled terms in the same cluster should, too*
  - example: the embedding for "black cherry cheesecake" is close to many embeddings that are "flavours"

Data cleaning module

- Sorts through existing and newly extracted data to see whether any of it was misclassified in the source texts
- ML model based on the Transformer architecture:
  - inputs:
    - textual product description
    - an attribute (flavor, volume, color, etc.)
    - a value for that attribute (chocolate, 16 ounces, blue, etc.)
  - output:
    - is the attribute value ok or not?
  - training data:
    - positive examples – valid and frequent attribute-value pairs for a product type (all ice cream types have flavors)
    - negative examples – replacing correct values with mismatched ones

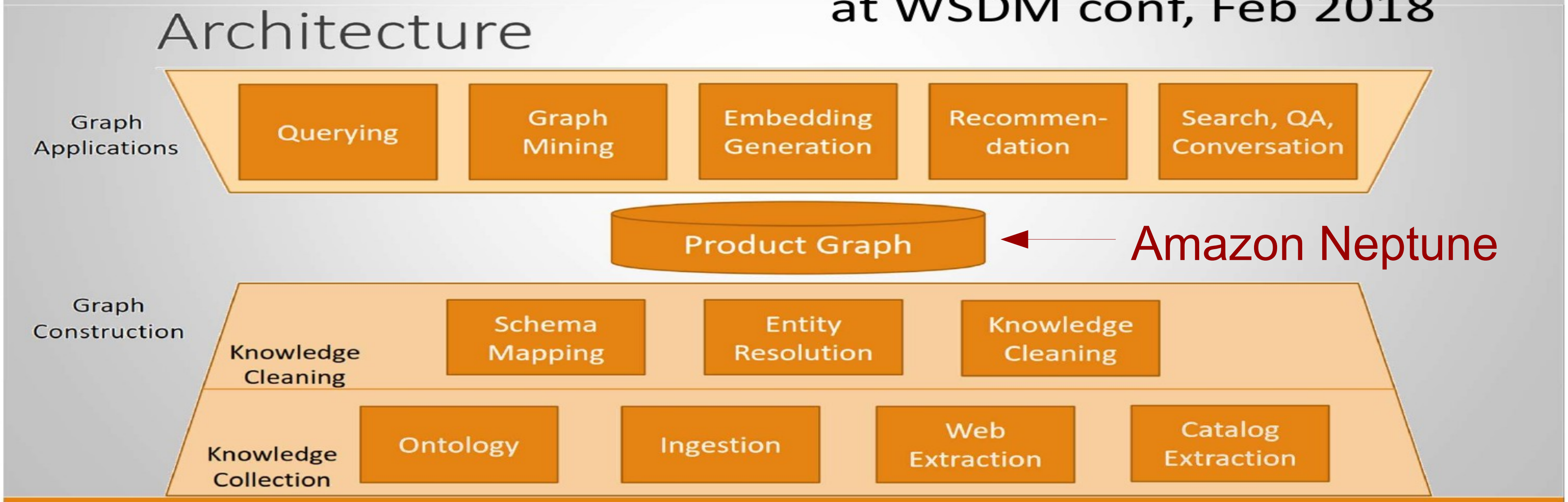
Synonym finding module

- Identify entity types and attribute values that have the same meaning
- Candidate synonyms:
  - analyse customer interaction data:
  - identify items that were viewed during the same queries
- Filter the candidates:
  - edit distance (a measure of the similarity of two strings of characters)
  - “a neural network”

Amazon

"We aim at building an authoritative knowledge graph for all products in the world"

Xin Luna Dong, Amazon, at WSDM conf, Feb 2018

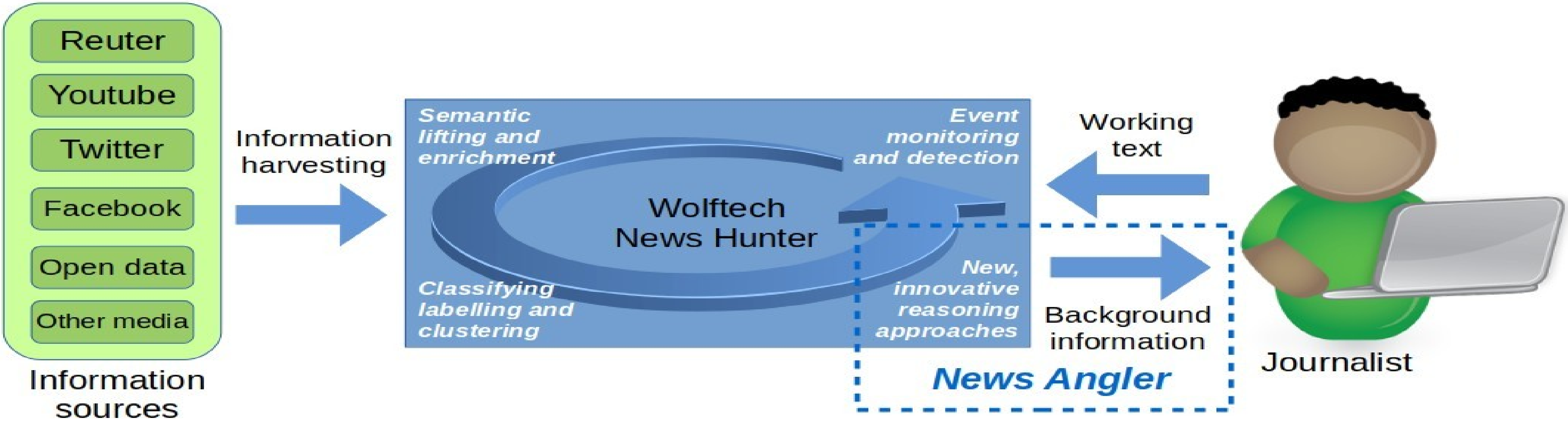


“Ongoing efforts”

- Handling products with multiple hypernyms
- Using image data in addition to structured data and NL text

The News Hunter Platform

Ongoing project: News Angler

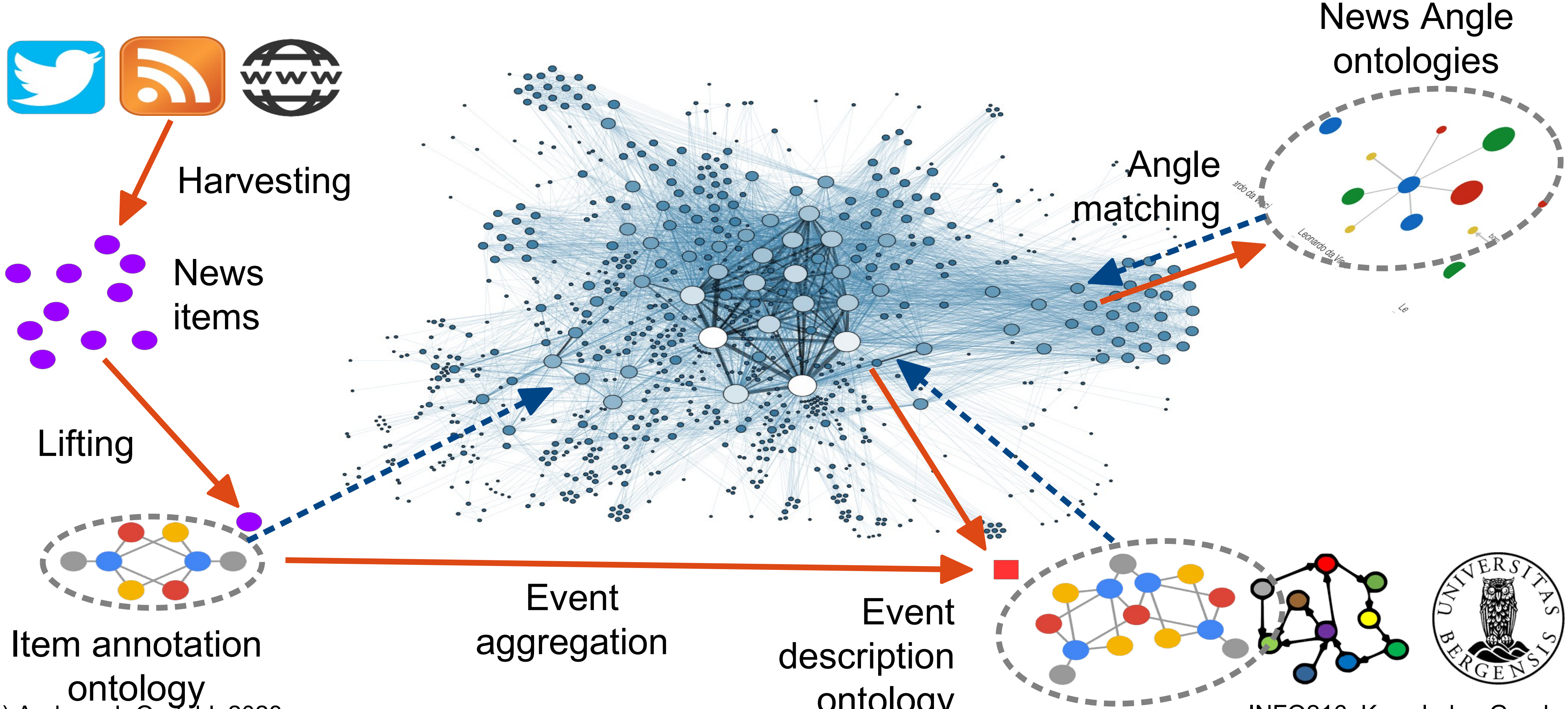


"Wolftech News supports and improves the workflows in a newsroom through mobile solutions for field work that are integrated with central systems for news monitoring, resource management, news editing, and multi-platform publishing"

1) Harvesting and analysing messages
2) Growing a semantic news graph
   - concepts, named entities, context…
3) Analysing working texts (stories)
4) Identifying background information
5) Prioritising and preparing
6) Journalistic and editorial preferences

Research: graph, searches, preparation, preferences, language, scaling

A single central news graph



Services

- Written in Python 3.8-3.9
- All services are deployed in docker containers
- FastAPI as the main python library for writing APIs

Services - harvesters

- Twitter harvester: connects to the Twitter API to read streams of tweets from news organizations accounts
- RSS harvester: downloads RSS feeds from news organisations
- GDELT harvester: gets the events and GKG datasets from GDELT projects
- NewsAPI harvester: use NewsAPI.org API to get real-time feeds of news from thousands of news outlets

Services - lifters

Lifters for news and GDELT that use NER to represent the information into knowledge graphs

- DbpediaSpotlight NEL: using DBpediaSpotlight for named entity linking
- SpaCy NEL: using SpaCy for named entity linking
- Kolitsas NEL: using Kolitsas algorithm for named entity linking

Technologies

- Docker Swarm
- Kafka (as pub/sub message queue to communicate between all services in the platform)
- Zookeeper
- Cassandra (storing raw data in a distributed cluster)
- Blazegraph (knowledge graph of news and events)
- MongoDB (configuration and metadata)
- All of them have been deployed using Docker containers

News Hunter Platform:
- **38 vCPUs**
- **152GB RAM**
- **20TB Disk**
- **17 Instances**

+

1 Launcher instance for deploying the cloud infrastructure:
- **1 vCPU**
- **4 GB RAM**

1 vCPU = 0.5CPU

Slide by Marc Gallofré Ocaña

JSON / JSON-LD

JSON

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

Name-Value pair

Object

Name

Value (String, Number, Boolean, null or Array)

JavaScript Object Notation (JSON)
www.json.org

JSON

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

Name-Value pair

Object

Name

Value (String, Number, Boolean, null or Array)

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name":
        {
          “firstname” : "Markus",
           “lastname” : “Lanthaler”
        },
  "workplaceHomepages": [ "http://www.tugraz.at/", “http://www.uib.no” ]
}
```

Array

Element (String, Number, Boolean, null or Object)

JSON

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

Name-Value pair

Object

Name

Value (String, Number, Boolean, null or Array)

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name":
       {
         “firstname” : "Markus",
         “lastname” : “Lanthaler”
       },
  "workplaceHomepages": [ "http://www.tugraz.at/", “http://www.uib.no” ]
}
```

Array

Element (String, Number, Boolean, null or Object)

Almost identical to Python dictionaries and lists!

JSON-LD

JSON

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

This is the person's id!

http://xmlns.com/foaf/0.1/name

http://xmlns.com/foaf/0.1/workplaceHomepage

http://xmlns.com/foaf/0.1/Person

How to represent semantic data in JSON?

JSON-LD

This is the person's id!

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

http://xmlns.com/foaf/0.1/name

http://xmlns.com/foaf/0.1/workplaceHomepage

http://xmlns.com/foaf/0.1/Person

JSON Linked Data (JSON-LD)
json-ld.org

```
{
  "@id": "http://me.markus-lanthaler.com",
  "@type" : "http://xmlns.com/foaf/0.1/Person",
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
          { "@id" : "http://www.tugraz.at/" }
}
```

JSON-LD to Turtle

JSON Linked Data (JSON-LD)
json-ld.org

```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
          { “@id” : "http://www.tugraz.at/" }
}
```

JSON-LD to Turtle

<http://me.markus-lanthaler.com>

JSON Linked Data (JSON-LD)
json-ld.org

```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
          { “@id” : "http://www.tugraz.at/" }
}
```

JSON-LD to Turtle

```
<http://me.markus-lanthaler.com>
        a        <http://xmlns.com/foaf/0.1/Person> ;
```

JSON Linked Data (JSON-LD)
json-ld.org

```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
          { “@id” : "http://www.tugraz.at/" }
}
```

JSON-LD to Turtle

```
<http://me.markus-lanthaler.com>
        a          <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/name>
                   "Markus Lanthaler";
```

JSON Linked Data (JSON-LD)
json-ld.org

```
{
  "@id": "http://me.markus-lanthaler.com",
  "@type" : "http://xmlns.com/foaf/0.1/Person",
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        { "@id" : "http://www.tugraz.at/" }
}
```

JSON-LD to Turtle

```
<http://me.markus-lanthaler.com>
        a            <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/name>
                     "Markus Lanthaler”;
        <http://xmlns.com/foaf/0.1/workplaceHomepage>
                     <http://www.tugraz.at/> .
```

JSON Linked Data (JSON-LD)
json-ld.org

```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
          { “@id” : "http://www.tugraz.at/" }
}
```

JSON-LD to Turtle

```
<http://me.markus-lanthaler.com>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/name>
                "Markus Lanthaler";
        <http://xmlns.com/foaf/0.1/workplaceHomepage>
                "http://www.tugraz.at/" .
```

JSON Linked Data (JSON-LD)
json-ld.org

```
{
  "@id": "http://me.markus-lanthaler.com",
  "@type" : "http://xmlns.com/foaf/0.1/Person",
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        "http://www.tugraz.at/"
}
```

Some reserved keys in JSON-LD

- @id: the JSON object with the @id key is identified by a particular URI
- @type: the JSON object with the @type key has a particular RDF type (or several types)
- @value: a value is a literal
- @context: a JSON object that contains the context (or semantic mapping) for the other objects in the same JSON array
- @base, @graph, @language, @vocab, ...

JSON-LD context

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/homepage",
      "@type": "@id"
    }
  },
  "@id": "http://me.markus-lanthaler.com/",
  "name": "Markus Lanthaler",
  "homepage": "http://www.markus-lanthaler.com/"
}
```

JSON-LD forms

- The same graph can be expressed in different ways:
  - *expansion* removes context by pushing semantics out into the objects
    - also does regularisation
  - *compaction* simplifies the objects by pulling semantics back into the context
  - *flattening* creates a normalised form for easier parsing by computer
- Regularised and normalised forms are easier to program than “free” JSON-LD because they have a more consistent structure

Final session:
Questions and quizzes

Wednesday May 31st

1215-1400