# Deep and Broad NLP for Big Data and Knowledge Graph Generation

Damir Cavar

March 2019

# NLP-Lab

https://nlp-lab.org/

Aarnav, Abhishek Babuji, Oren Baldinger, Rohit Bapat, Shantanu Bedeker, Aarushi Bisht, Jagpreet Singh Chawla, Boli Fang, Maanvitha Gongalla, Peace Han, Murali Kishore Varma Kammili, Anurag Kumar, Mahesh Latnekar, Shujun Liu, Umang Mehta , Prashant Modak, Shreejith Panicker, Chaitanya Patil, Gopal Seshadri, Richard Xu, Yiwen Zhang, …

# Agenda

- NLP technologies
    - RESTful Microservice Infrastructure
- JSON-NLP
- HooSIER Text to Graph
- Research Directions

# Motivation for NLP Infrastructure and API

- NLP Interoperability
- NLP Complexity
- NLP Errors

# Motivation

- NLP Interoperability
  - Syntactic level: output formats are incompatible
  - Semantic level: annotation standards do not exist
  - Incompatibility of outputs and annotations

- Example:
  - Stanford CoreNLP
  - FreeLing
  - Spacy
  - …

# Issues

- NLP Complexity:
  - Interpretation and processing of outputs
    - Structural information
    - Meaning of Part-of-Speech tags
    - Interpretation of Dependency Labels

- Example:
    - Constituent Parse Tree
    - Lexical Functional Grammar C- and F-structure

# NLP Issues

- NLP Errors in analysis:
  - Expert knowledge and knowledge of language necessary
  - Model specific error types

- Example:
  - Allen NLP:
    - Coreference Resolution
    - Constituent Parser
    - Dependency Parser
    - Open Domain Information Extraction (OpenIE)
  - Stanford CoreNLP
  - …

# NLP Infrastructure

- NLP Errors
  - Introduce Redundancy: Multiple NLP components for the same annotation task
  - Repair systematic output errors
- NLP Complexity
  - Build an API to simplify access, facilitate use of advanced NLP output
- NLP Interoperability
  - Normalization and standardization of output formats and annotations
  - Uniform API for NLP

# NLP Services

- Functional Aspects
  - Differences in Linguistic Annotation
  - Underlying Models/Theories Differ
- Solution
  - Merging outputs from different NLP components

# NLP Services

- Technical Issues
  - Configurational Complexity
  - Dependencies for libraries, modules, extensions (Python, Java, C++)
  - Memory
    - Large models
    - Runtime memory requirements
    - Storage (file, db) requirements
  - Platform limitations
  - Hardware requirements: CPU & GPU

# Microservice Architecture

- Solution: RESTful Microservice architecture
  - Scalability
  - Target platform and remote access (intranet or Cloud service)
  - Flexibility
    - Replaceable components
    - Versioning
  - Open to numerous programming languages, systems, architectures
    - Dominance of Python in NLP limits engineering possibilities and integration of NLP in larger production environments

# NLP Services

- Most commonly used NLP services:
  - Tokenizer
  - Sentence segmentation
  - Part-of-Speech Tagging
  - Embeddings
- Less common NLP services:
  - Morphological analysis
  - Coreference and Anaphora Resolution
  - Dependency Parsing
  - Constituent Parsing
  - Semantic Role Labeling
  - …

# NLP API

- Linguistic complexity as a barrier
  - Understanding of Parse Trees and Potential Use in Applications

- API as a Translational Service
  - Mapping of Linguistic Information to Useful Services
  - Transformation of NLP Output

- Example:
  - Scope relation and syntactic trees
  - Part-of-Speech tags and morph-syntactic features

# NLP Output Format

- Normalization and Interoperability via Output Standardization
- Other Standards (no real standards)
    - CONLL – text-based line to token format
    - Proprietary JSON and XML formats
    - Binary objects in Python
- Issues with other standards:
    - Lack of interoperability
    - Lack of features
    - Data size and processing complexity

# JSON-NLP

- JSON:
  - Full support in most important programming languages
  - Human readable
  - Compact and efficient
- Extended normalized feature set:
  - Document level annotation: meta-info, tokens
  - Annotation of e.g. coreference types, semantic, pragmatic features
  - Translational layer: making implicit features explicit, providing features like voice, tempus, aspect
  - Annotation of discontinuities
  - Implicit, covert tokens (e.g. ellipsis, gapping, implicit arguments)

# JSON-NLP

- Open and Free on GitHub (https://github.com/dcavar/JSON-NLP)
- Converters from major NLP pipelines and components to JSON-NLP
- Converters from JSON-NLP to other formats (e.g. CONLL), lossy conversion
- JSON Schema with Validation
- Translation of NLP output to extended annotations in JSON-NLP
- Enabling:
  - Middleware for NLP for abstraction
  - NLP output comparison

# JSON-NLP

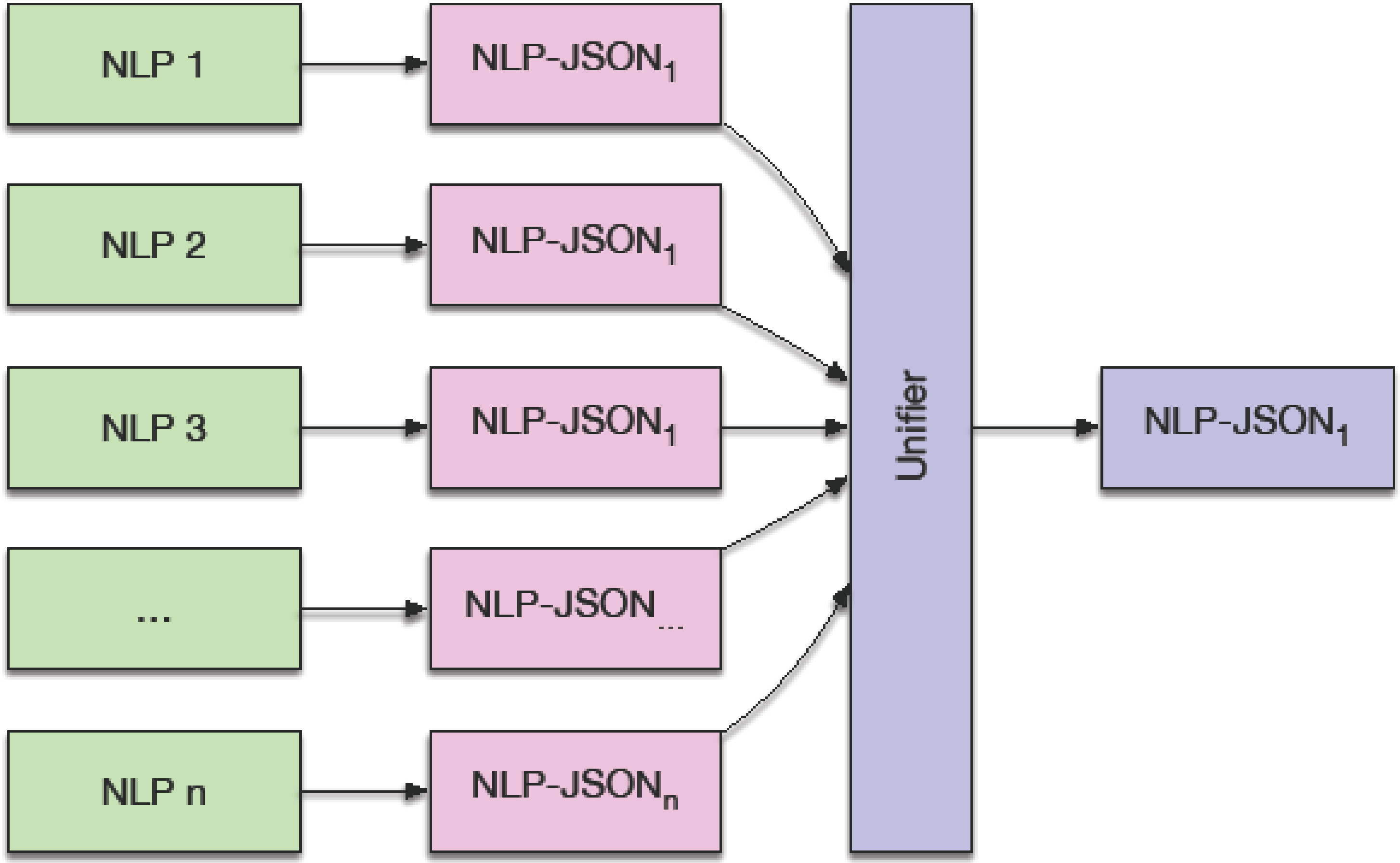
- Extended features
  - Encoding of time reference, duration of events, prosody, intonation, focus
  - Clause detection, identification of phrasal heads and compounds

- Unification
  - Symbolic and Probabilistic Algorithm
  - Merging of *n*-JSON-NLP files
  - Detection of mismatches in NLP-annotations

- Facilitates
  - Deeper comparison and evaluation of individual NLP components
  - Ensembles of NLP components or pipelines

# NLP Ensemble

- HooSIER
  NLP-ensemble



# NLP Infrastructure

- Python-based technologies
  - spaCy, Flair, Polyglot, Natural Language Toolkit (NLTK), Xrenner, …
- Java-based technologies
  - OpenNLP, LingPipe, Stanford CoreNLP, Malt Parser, …
- Hybrid technologies
  - E.g. C(++) Foma in Java with JNI, in Python
- Included models:
  - Word embeddings: word2vec, GloVe, Numberbatch, FastText, Flair, ELMo, BERT
- All available as: RESTful Microservices with JSON-NLP output

# NLP Infrastructure

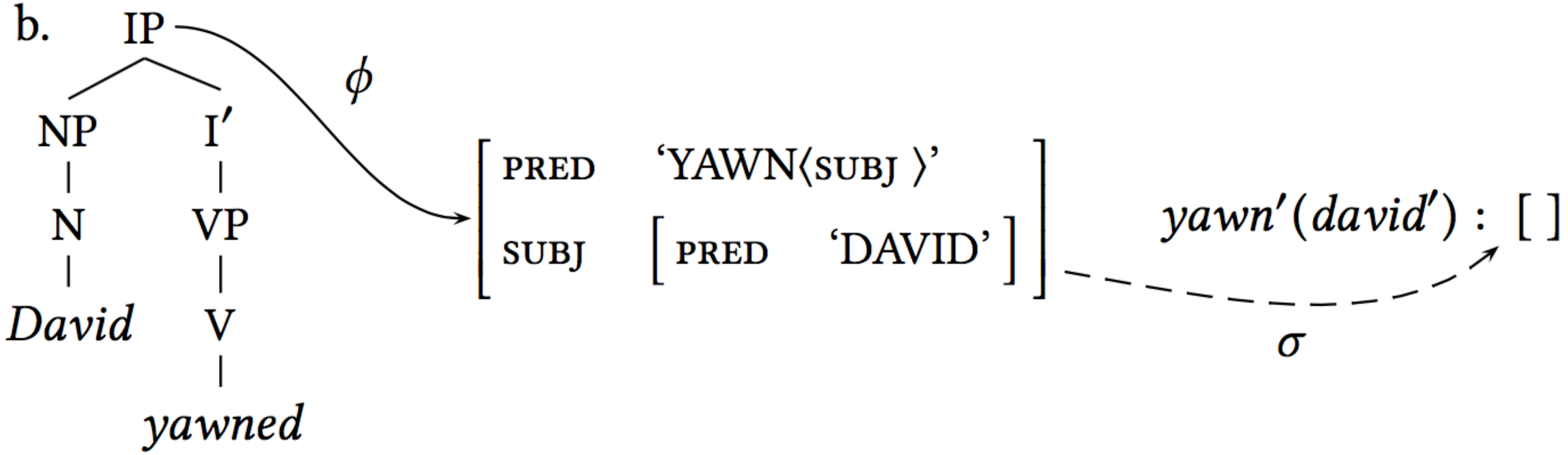
- Facilitating research
  - Evaluation and comparison of NLP components, models, embeddings
  - Ensembles of NLP components solving problems that cannot be solved end-to-end using Deep Learning alone
    - Example: Coreference and anaphora resolution with semantic relevance
      - Take the knife, cut the lime in two halves, and put it down.
      - Take the knife, cut the lime in two halves, and squeeze it.
  - Generating ambiguities to work around lack of interactive parallelism:
    - *John met Peter. He likes him a lot.*
    - *He* could be *John* and *him* could be *Peter* or *He* could be *Peter* and *him* could be *John* or …
  - Deep NLP for any kind of text or language processing

# Semantic Processing

- Meaning and Compositionality as Formal Mapping from Syntax to Semantic Representation (Bresnan, LFG)

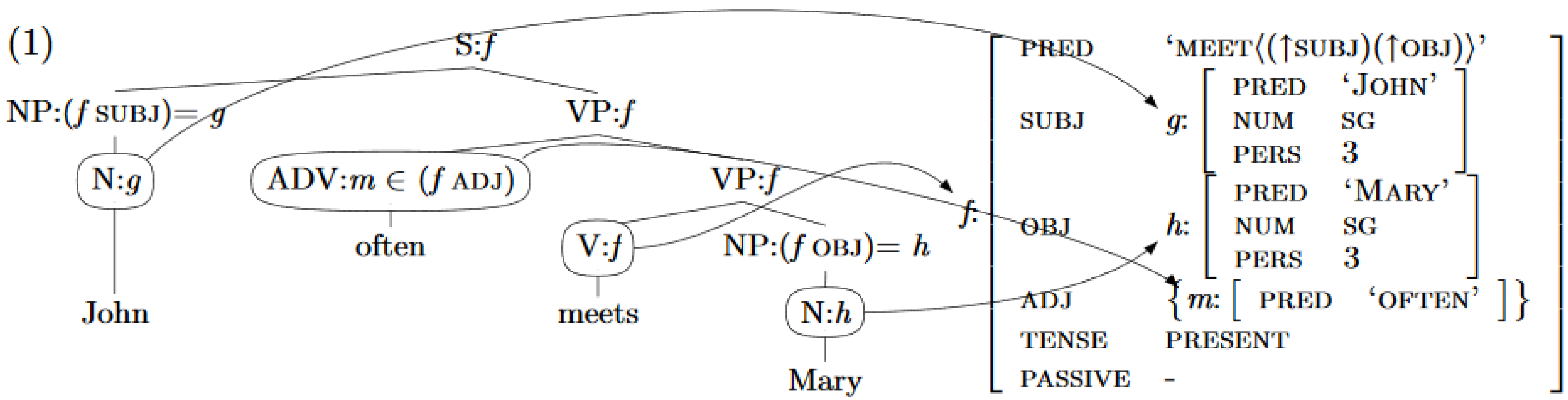
a. David yawned.

b.



Frank & Genabith (2001) Glue Semantic Analysis

(1)




**Lexical entries with associated meaning constructors**

John N $(\uparrow \text{PRED}) = \text{‘JOHN’}$
$john : \uparrow_\sigma$

Mary N $(\uparrow \text{PRED}) = \text{‘MARY’}$
$mary : \uparrow_\sigma$

meets V $(\uparrow \text{PRED}) = \text{‘MEET’}$
$\lambda y, x.meet(x,y) \colon (\uparrow \text{OBJ})_\sigma \multimap ((\uparrow \text{SUBJ})_\sigma \multimap \uparrow_\sigma)$

often ADV $(\uparrow \text{PRED}) = \text{‘OFTEN’}$
$\lambda P, x.often(P(x)) \colon ((\text{ADJ} \in \uparrow)\ \text{SUBJ})_\sigma \multimap (\text{ADJ} \in \uparrow)) \multimap ((\text{ADJ} \in \uparrow)\ \text{SUBJ})_\sigma \multimap (\text{ADJ} \in \uparrow))$

**Instantiated meaning constructors**

$john : g_\sigma$
$mary : h_\sigma$
$\lambda y, x.meet(x,y) \colon h_\sigma \multimap (g_\sigma \multimap f_\sigma)$
$\lambda P, x.often(P(x)) : (g_\sigma \multimap f_\sigma) \multimap (g_\sigma \multimap f_\sigma)$

**Meaning derivation**

$$\dfrac{\dfrac{\dfrac{\lambda y, x.meet(x,y) : h \multimap (g \multimap f) \quad mary : h}{\lambda x.meet(x, mary) : g \multimap f} \quad \lambda P, x.often(P(x)) : (g \multimap f) \multimap (g \multimap f)}{\lambda x.often(meet(x, mary)) : g \multimap f} \quad john : g}{often(meet(john, mary)) : f}$$

# Description Logic Approach

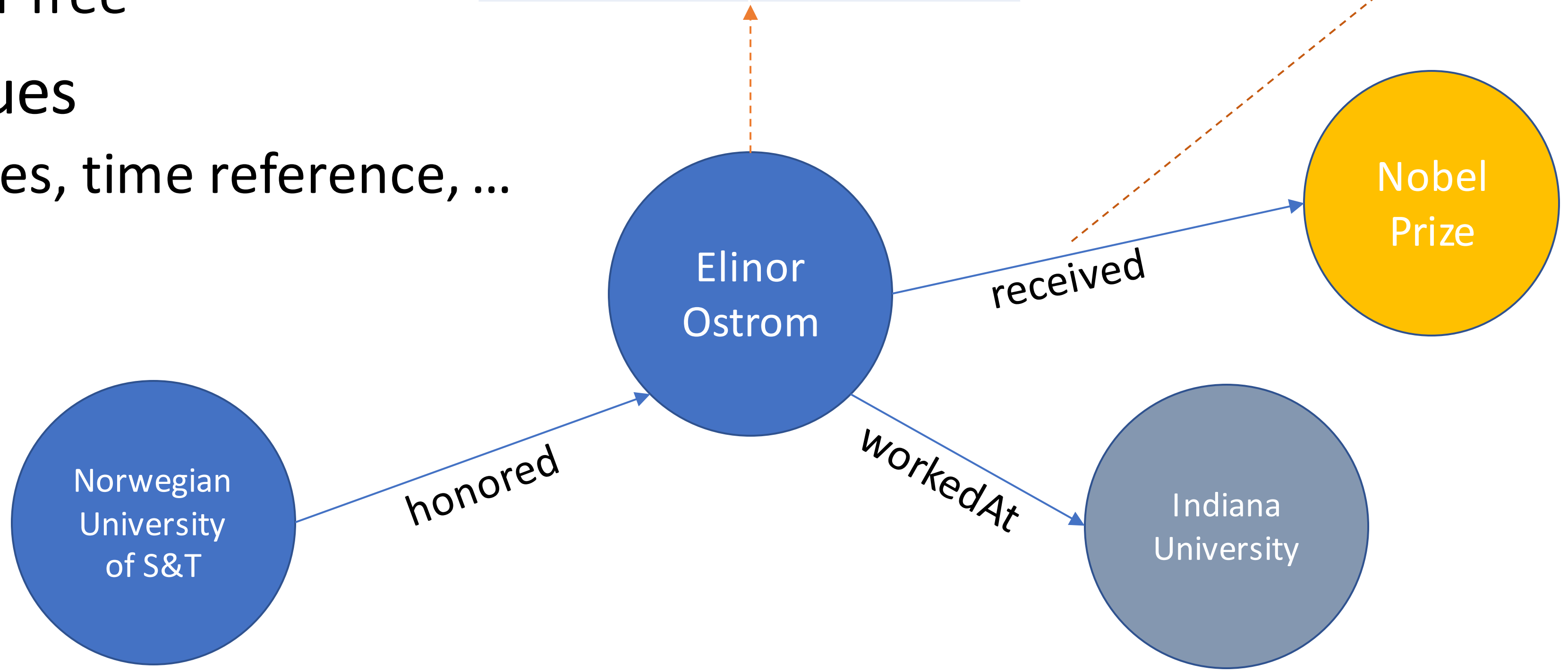
- Direct mapping of sentence and clause content to graph of concepts and relations
- Accumulating properties in concepts or nodes, and for relations or links:
  - Attribute-Value table
- OWL for semantic check and validity

# Knowledge Graphs

- Concepts and Relations
  - Mostly unconstrained
  - Domain specific or free
- Attributes and Values
  - encoding properties, time reference, …

| Attribute | Value |
|---|---|
| Birth place | LA, CA |
| DOB | 08/07/1933 |
| gender | female |
| … | |

| Attribute | Value |
|---|---|
| year | 2009 |
| … | |



# Semantic Relations

- Extraction of core semantic relations: predicate and arguments
- Example:
  - While travelling in Africa, John Smith, the CEO of Talora Inc. bought surprisingly a farm in Kenia.

| | | | | |
|---|---|---|---|---|
| John Smith | – | buy | – | a farm |
| PERS | | TRANS | | INDEF |
| SUBJ | | PRED | | OBJ |

- Required components
  - Deep NLP

# Information Extraction

- Basic NLP: tokenization, lemmatization, Part-of-Speech tagging, split into sentences
- More advanced: Clause level segmentation
- Parsing: Dependency and Constituent Structure

- Problems:
  - Margin of Error
- Solution:
  - Parallelization and NLP ensembles

# NLP Issues and IE

- Scope (missing in NLP technologies)
  - John bought a car.
  - Peter said that John bought a car.
  - It is not true that John bought a car.
- Ellipsis
  - John bought ~~a car~~ and Mary bought a car.
  - John bought a car and ~~John~~ drove to Canada.
- Gapping
  - John liked to read books and Mary ~~liked to read~~ newspapers.
- Implicit arguments:
  - John wants PRO to read a book.
  - Got it!

# NLP Extensions

- Knowledge representations
  - WordNet
  - VerbNet
  - PropBank
  - FrameNet
  - Knowledge Graphs
- Predict required arguments
- Extract advanced properties of concepts, predicates, events from knowledge representations
- Integrated in HooSIER NLP Infrastructure
  - Wrapped in JSON-NLP

# NLP Extensions

- Implicatures:
  - John to Peter: I bought the blue car.
    - John and Peter talked about cars earlier.
    - There should be a set with at least one more car the John could have bought, but did not, and
    - None of the cars in the set is blue.
  - Clues: Definiteness of NP via the, and specificity of NP
- Presuppositions:
  - John fed his cat this morning.
  - Assumptions:
    - John owns/has a cat/pet.
    - John owned cat-food this morning.
  - Clues: Possessive pronoun as modifier of Direct Object.

# Semantic Mapping and Reasoning

- Type of Predicative Arguments: Typing
  - Named Entity Recognition
  - Closes possible Hypernym in a Taxonomy or Ontology of isA relations

- Identity of entity: Linking
  - Named Entity Recognition
  - Link to unique identifier of entity in some knowledge representation, Ontology, Wikipedia, Knowledge Graph

- Issues: Ambiguity

# Linking Disambiguation

- Vector-based computation over text and graphs

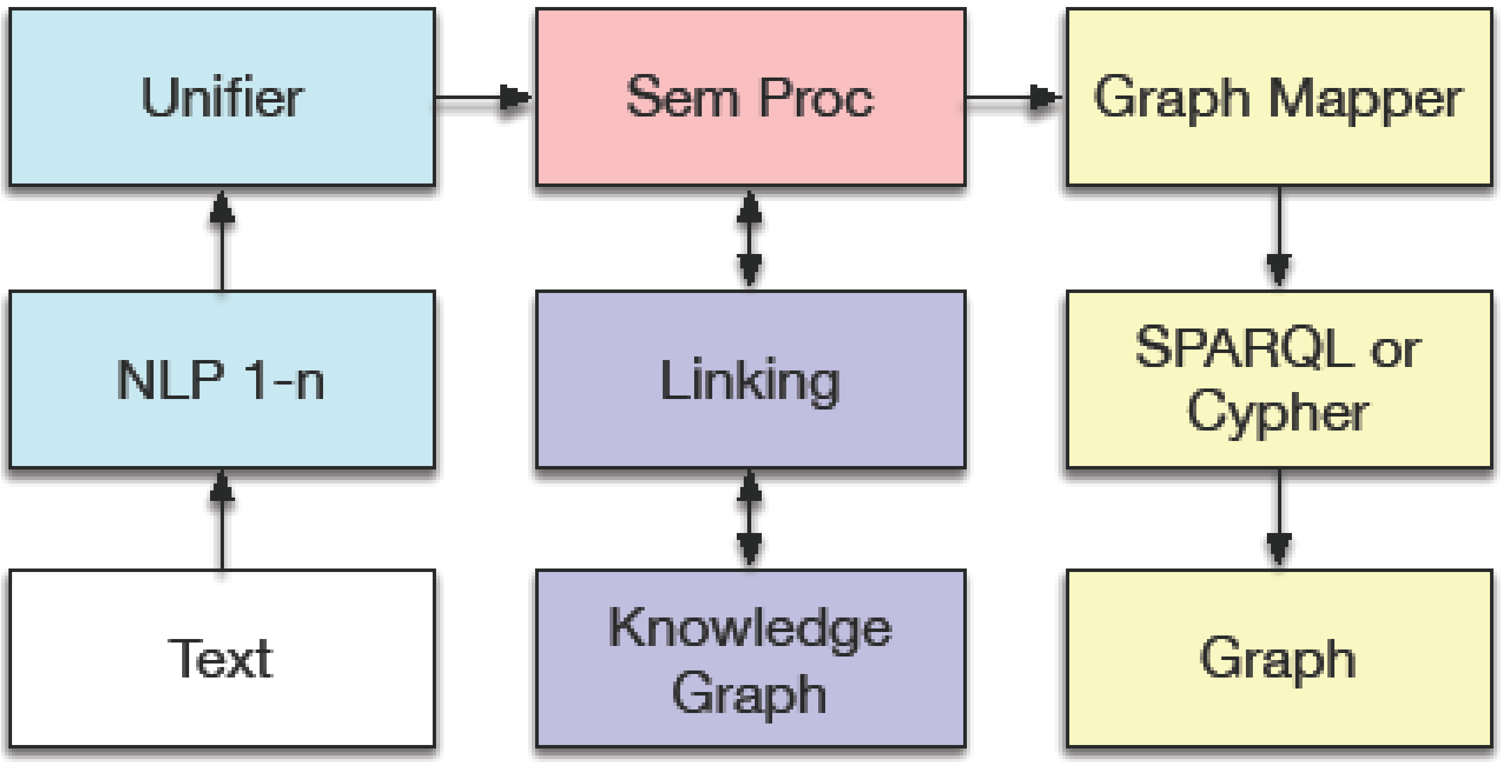
- Context prediction:
  - Compute the prediction of the context words in text with entity for all link-candidates

- Similarity:
  - Vectorize the sub-net of all concepts in Knowledge Graph and compute the similarity to the text with entity

- Example

# Pipeline

- Knowledge Graph Generation



# NLP Infrastructure

- Knowledge Graphs as RESTful Microservices
  - YAGO integrated in Apache Jena with TDB, SPARQL interface, Lucene index
  - ConceptNet using remote API
  - Microsoft Concept Graph via interface to MongoDB
  - DBpedia using remote API, possible setup as for YAGO
  - SPARQL-based n-hop search and string-similarity search (mutli-lingual)

- Generated Graphs
  - Neo4J using Cypher
  - Stardog using SPARQL
  - Open format based on abstract graph class

# Other systems

- FRED Graph Extraction
  - http://wit.istc.cnr.it/stlab-tools/fred/
  - http://wit.istc.cnr.it/stlab-tools/fred/demo/
- FreeLing
  - http://nlp.lsi.upc.edu/freeling/demo/demo.php

- Limitations:
  - NLP restricted
  - Graphs or Networks limited or restricted

# Research Directions

- Encoding of Events and Event Types using graphs
- Link Prediction or computation of Paradigmatic relations
- Network representation of typed concepts
- Forensic Research: with implicatures and presuppositions
- News article comparison
- Abstract semantic search over typed and linked concepts and entities
- Information validation, knowledge mapping
- Etc.

# Resources

- JSON-NLP and Wrappers on GitHub repo
- KG Linking Disambiguator
  - Graph storage and SPARQL interface: YAGO, ConceptNet, DBpedia, Microsoft Concept Graph
- NLP RESTful Microservice Modules (Java, Python, C(++))
  - JSON-NLP output conversion
  - RESTful wrappers for: Stanford CoreNLP, Apache OpenNLP, LingPipe, spaCy, Flair, Polyglot, NLTK, Xrenner, etc.

- Apache License 2.0

# NLP Infrastructure

- Estimated Server Requirements without stress-test
  - WildFly 16 and Java 11
  - Python 3.x
  - GPU recommended
  - Disk space for data, models, DBs: min. 2 TB (possibly more with DBpedia)
  - RAM for daemons, services, runtime: min. 128 GB