MuLLinG

MultiLevel Linguistic Graphs for Knowledge Extraction

Vincent Archer

i3S sophia antipolis

Université Nice Sophia Antipolis

cnrs

TextGraphs-5 (ACL 2010 Workshop) - 16 July 2010

Motivation

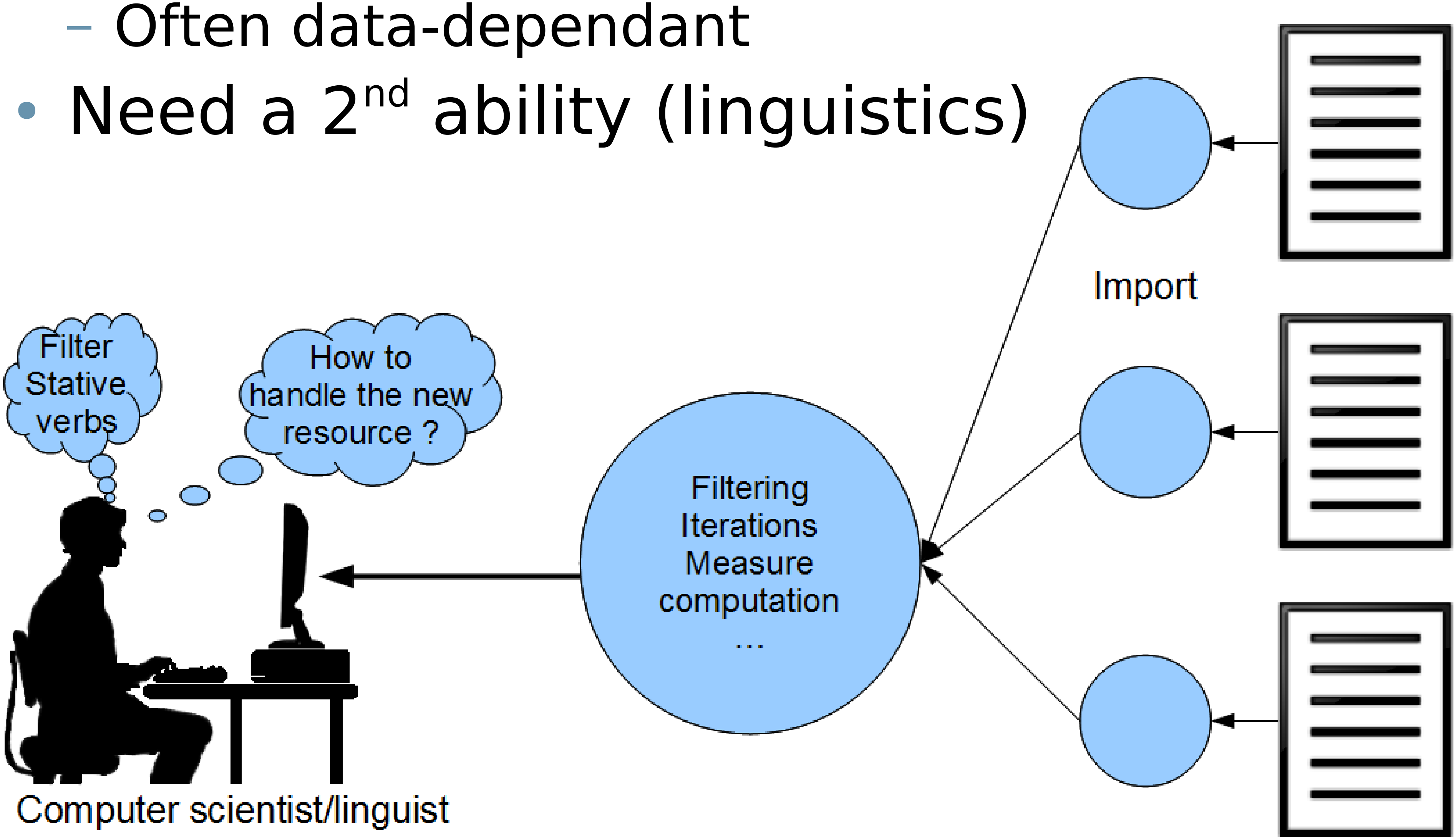
- NLP: Low-quality results
  - Ambiguities, particular phenomena
- Lack of linguistic (lexical) resources
  - Focus: extraction of lexical information
    - Computer scientist POV
  - Should be produced quickly and precisely
  - We must combine human and machine abilities
- Need: Generic tools

Outline

- Make lexical extraction easier
- MuLLinG model
- Application: collocation extraction

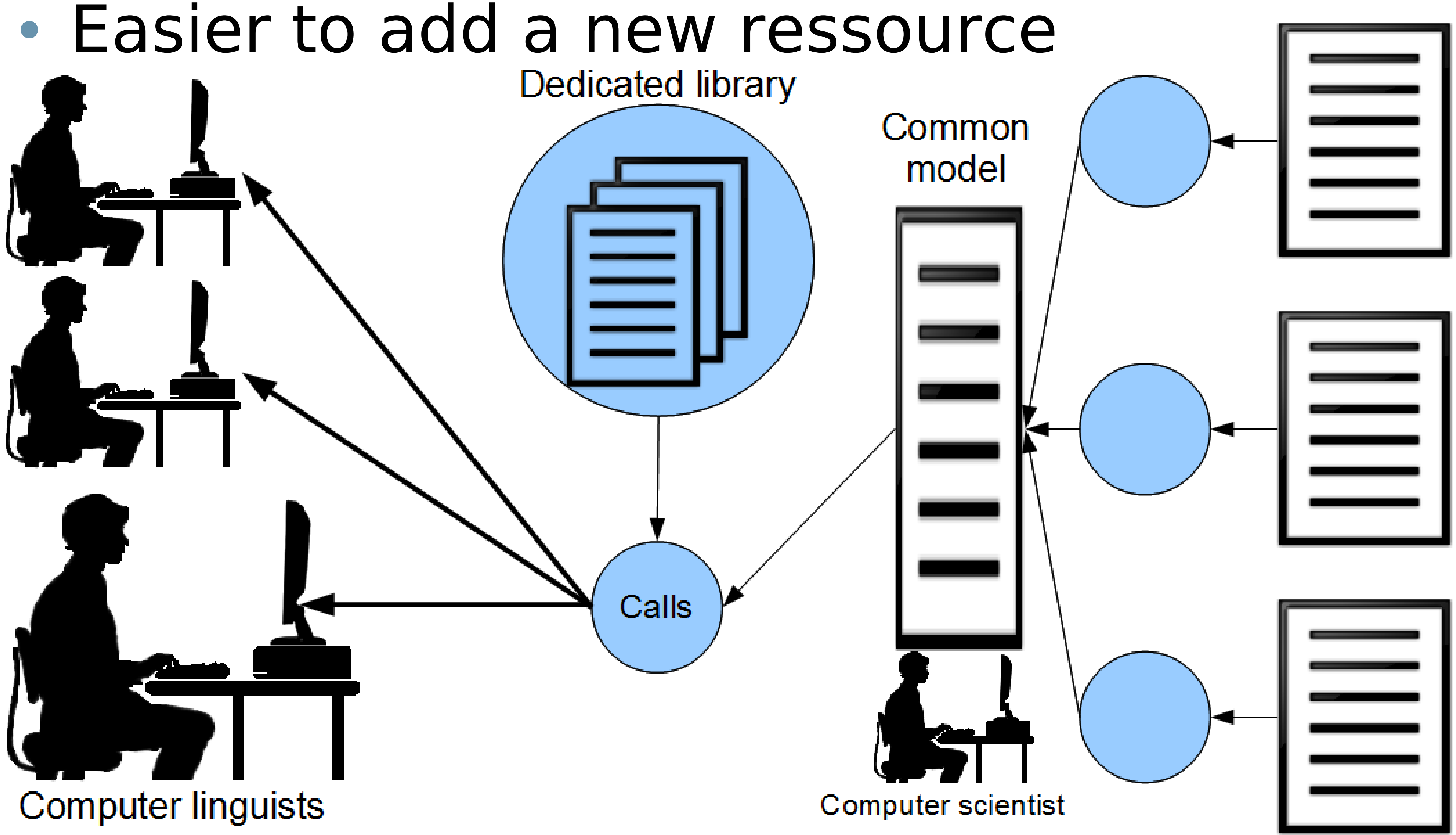
Difficult Programming

- Resource management (add a new one?)
  - Often data-dependant
- Need a 2nd ability (linguistics)


Import
Filter Stative verbs
How to handle the new resource ?
Filtering Iterations Measure computation …
Computer scientist/linguist

Separation of tasks – Generic tools

- Requires less knowledge in programming
- Easier to add a new ressource

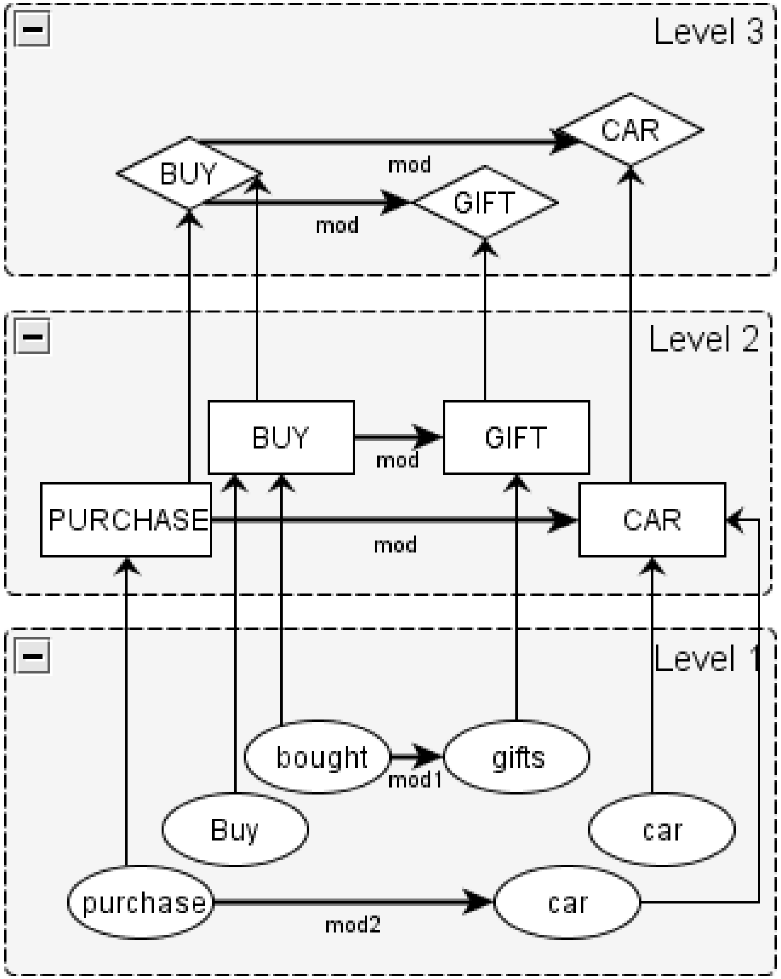


What kind of model ?

- Simple representation of data
  - Expressive (able to model complex data)
  - Generic (corpora, dictionaries, etc.)
- Simple generic operations
  - Task- and data-independant
  - High-level
  - Combine simple operations, rather than write a complex one
- => Graphs
  - Relations (juxtaposition, dependency, etc.)
  - Easy to understand/handle, widely used in NLP

MuLLinG: MultiLevel Linguistic Graph

- Levels=different views
- Grouping by *equivalence classes*
  - 1 class = 1 node at superior level
    - Level hierarchy
  - Interlevel edges
    - Between a node and its class
- Attributes are free
  - No constraint on the data

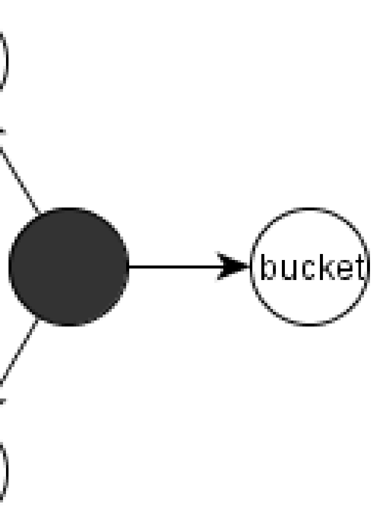


Associated operations

- Modifying the graph:
  - Parameters: level, filtering function, attribute computation functions (for nodes/edges)
    - Given by the user
  - *Emergence* creates a new level
- Union, intersection, difference of graphs
  - Based on identity of nodes/edges
- Basic:
  - Add/delete edge, node (and its descent)
  - Conditional application
  - Measure computation

Complex version

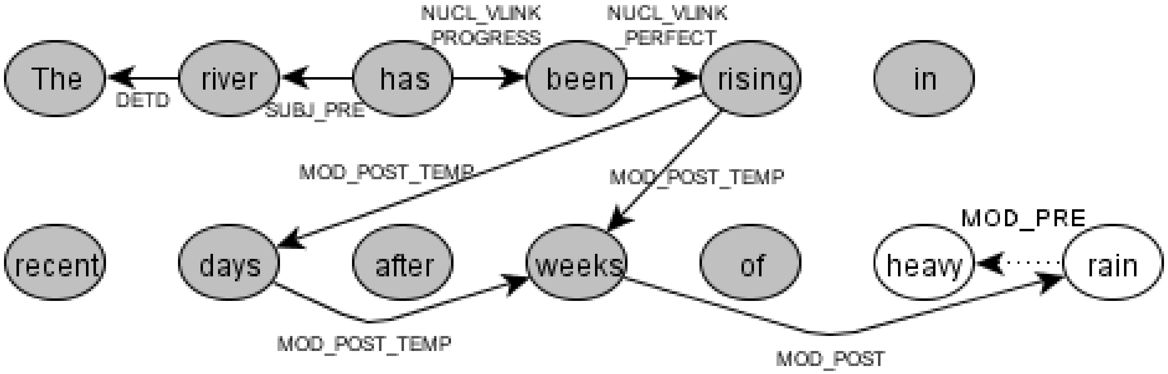
- Relations not always binary
- 1 relation =
  - 1 (standard) node materializing the relation
  - + numbered argument edges
- (operators are adapted)



Experiment: collocation extraction

- Collocation (« *Driving rain* ») = semi-fixed
  - One term is chosen arbitrarily
    - In function of the other one
    - To express a particular meaning
  - Problem for translation

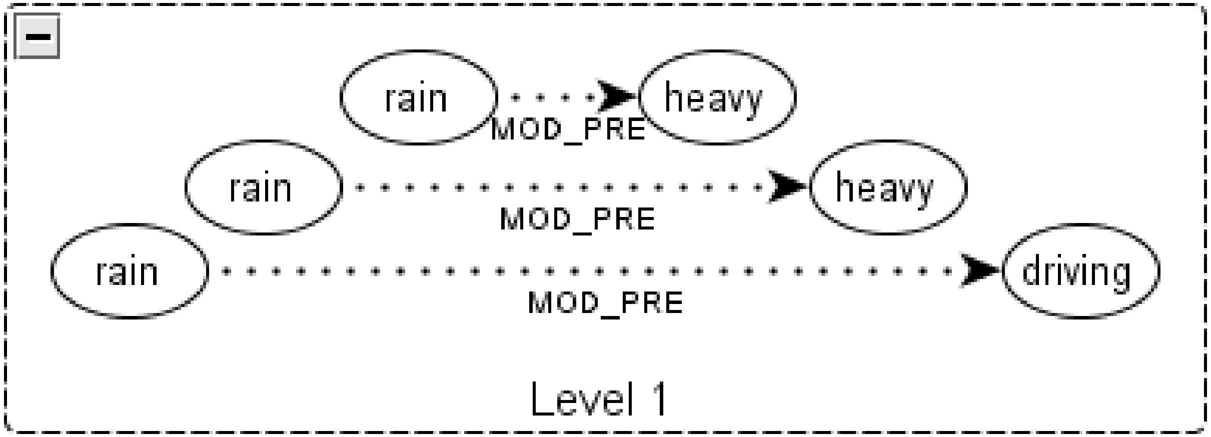
- Initial graph:
  - Dependencies produced by the parser (XIP)



Collocation extraction

- *Emergence* produces the superior level
  - Operation based on equivalence classes
    - Before: relations between objets
    - After: (grouped) relations between grouped objects
  - Parameters:
    - Level, filter, attribute computation
    - + function identifying the class of a node/edge

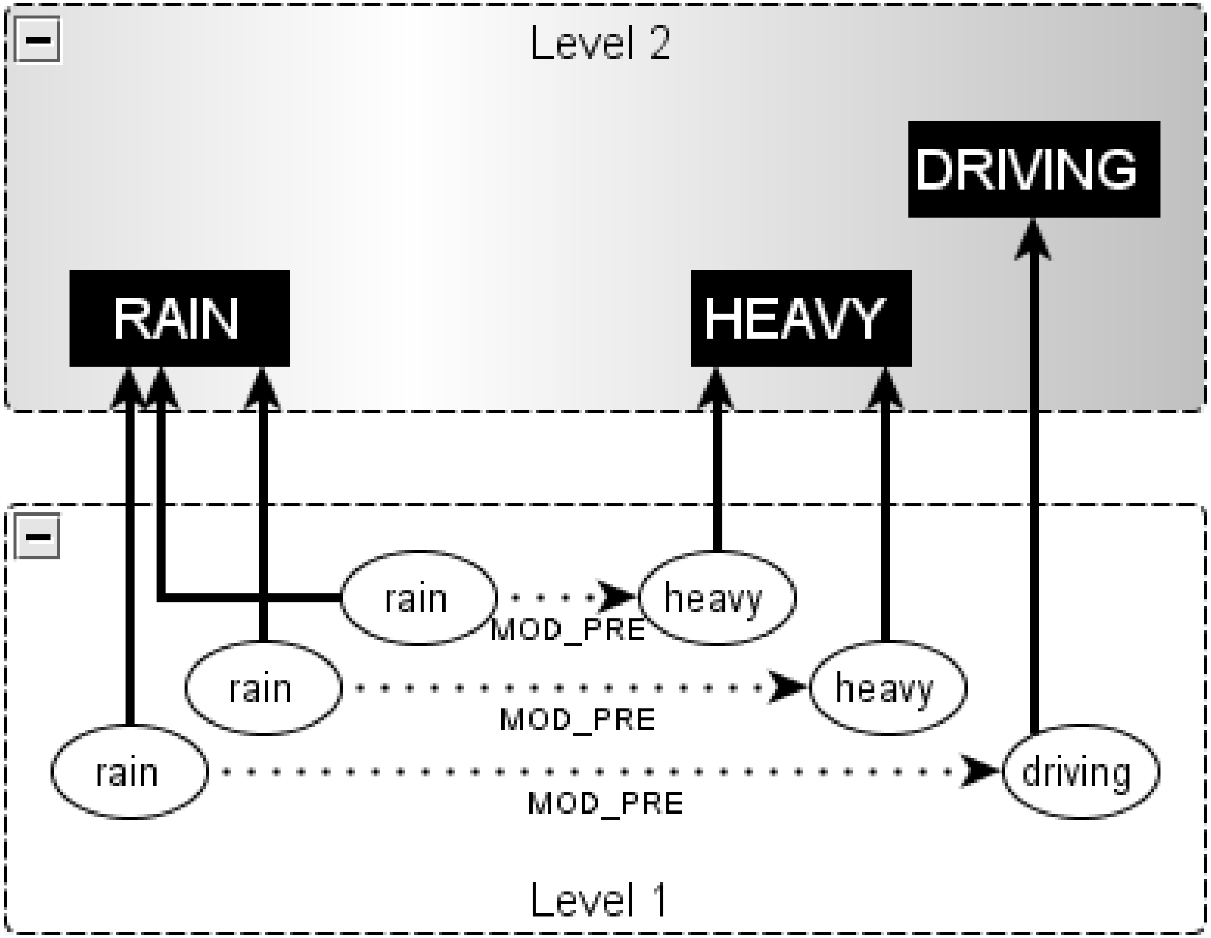
- Filtering relations:
  - Removing nodes



Collocation extraction
Node emergence

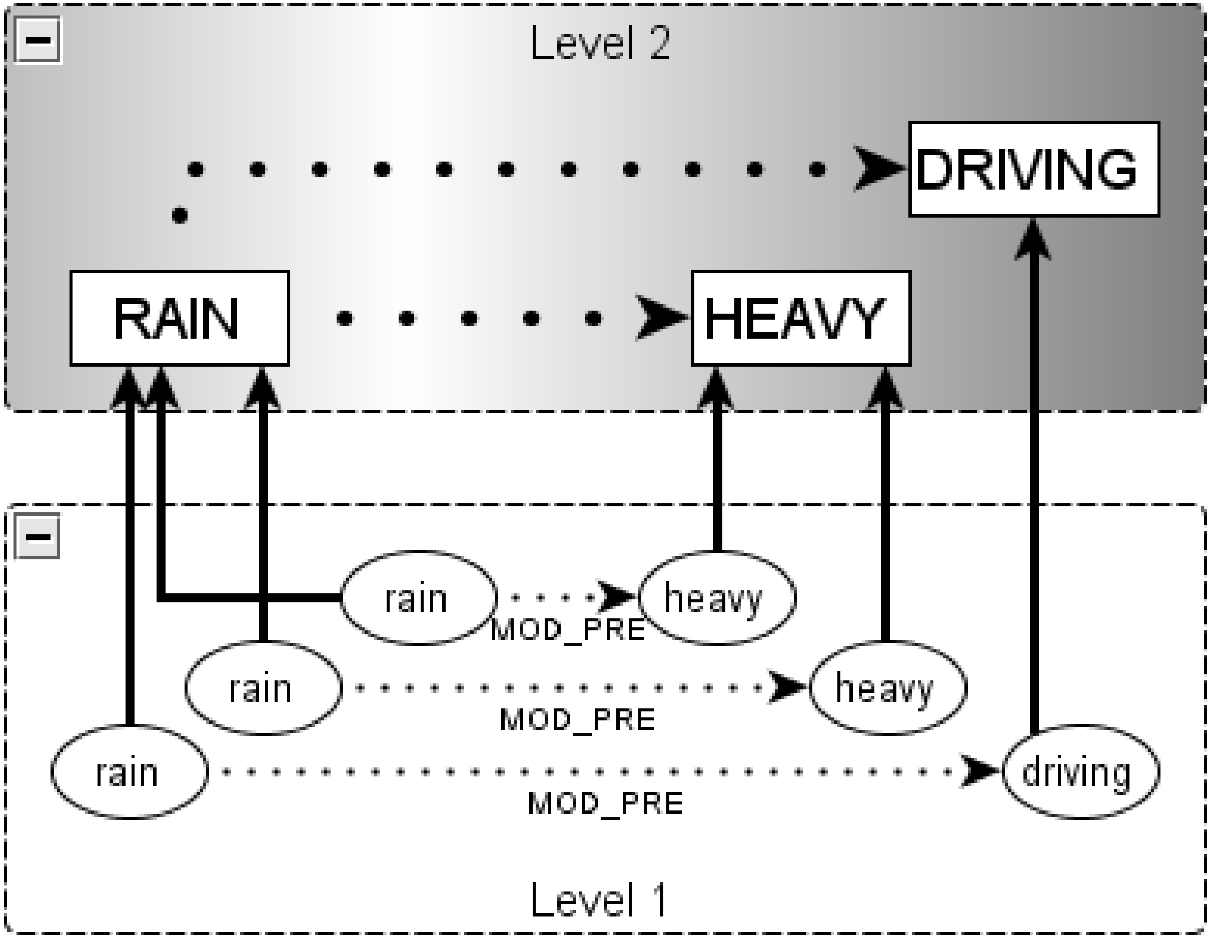
- 1 node = 1 equivalence class
  - Linked to nodes (at inferior level) elements of the class

```
NodeEmergence(
1,
true,
Class_Lemma_pos(),
CompAttrNode()
    //<id, idclass>
      <type, "term">
      <nboccs, incr>
)
```



Collocation extraction
Edge emergence

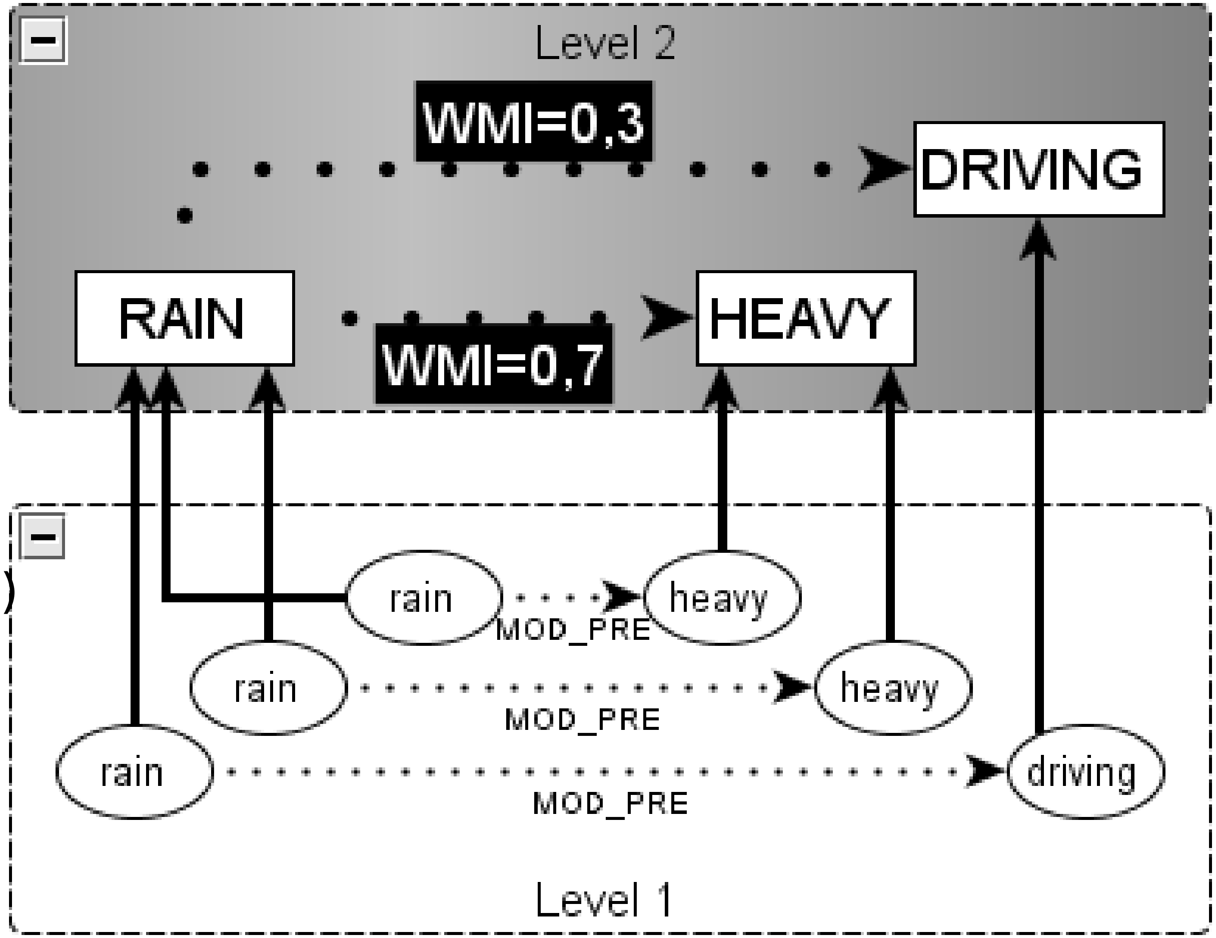
- 1 edge between A and B = 1 set of edges
  - Between an element from class A, and an element from class B
  - Equivalent



```
EdgeEmergence(
1,
true,
Class_Type()
CompAttrEdge(),
    //<id, idclass>
      <type, "classmod">
      <nboccs, incr>
CompAttrSource(),
    //<d+, incr>
CompAttrTarget() )
    //<d-, incr>
```

Collocation extraction Measure computation

- Using values previously computed
  - Number of (co-)occurrences, in/out degree...
- Candidates : level2 edges



```
ComputeMeasure(
WMI(),
    //association measure
is_classmod(),
"measure",
    //where to write (result)
"nboccs",
"nboccs",
    //where to read
NumberSentences
...)
```

Observations

- Coherent results
- Mulling (open-source C++ library)
  - http://mulling.ligforge.imag.fr
    - In/out file format: GraphML
  - **~70 lines** (calls)
    - vs. *Ad hoc*: **~400 lines** (iterations on the data)
    - Much faster description / Avoid programming errors
    - Import : ~250 lines (vs. 200 lines ad hoc)
  - Execution quite slower
    - less optimized
  - Generic: reusable with any kind of relation

Future works

- Library usability
  - Import
  - High-level (request) language
  - Graphic interface
  - Memory: use databases (+cache) to store large graphs
- Graph clustering
- Applications to other graphs
  - Less NLP-centered
  - Semantic web (RDF/SPARQL)
  - Social networks