DaSe Lab

Leveraging Wikipedia for Ontology Pattern Population from Text

AAAI-MAKE 2019

Stanford University

Michelle Cheatham, James Lambert
Charles Vardeman II

WRIGHT STATE UNIVERSITY

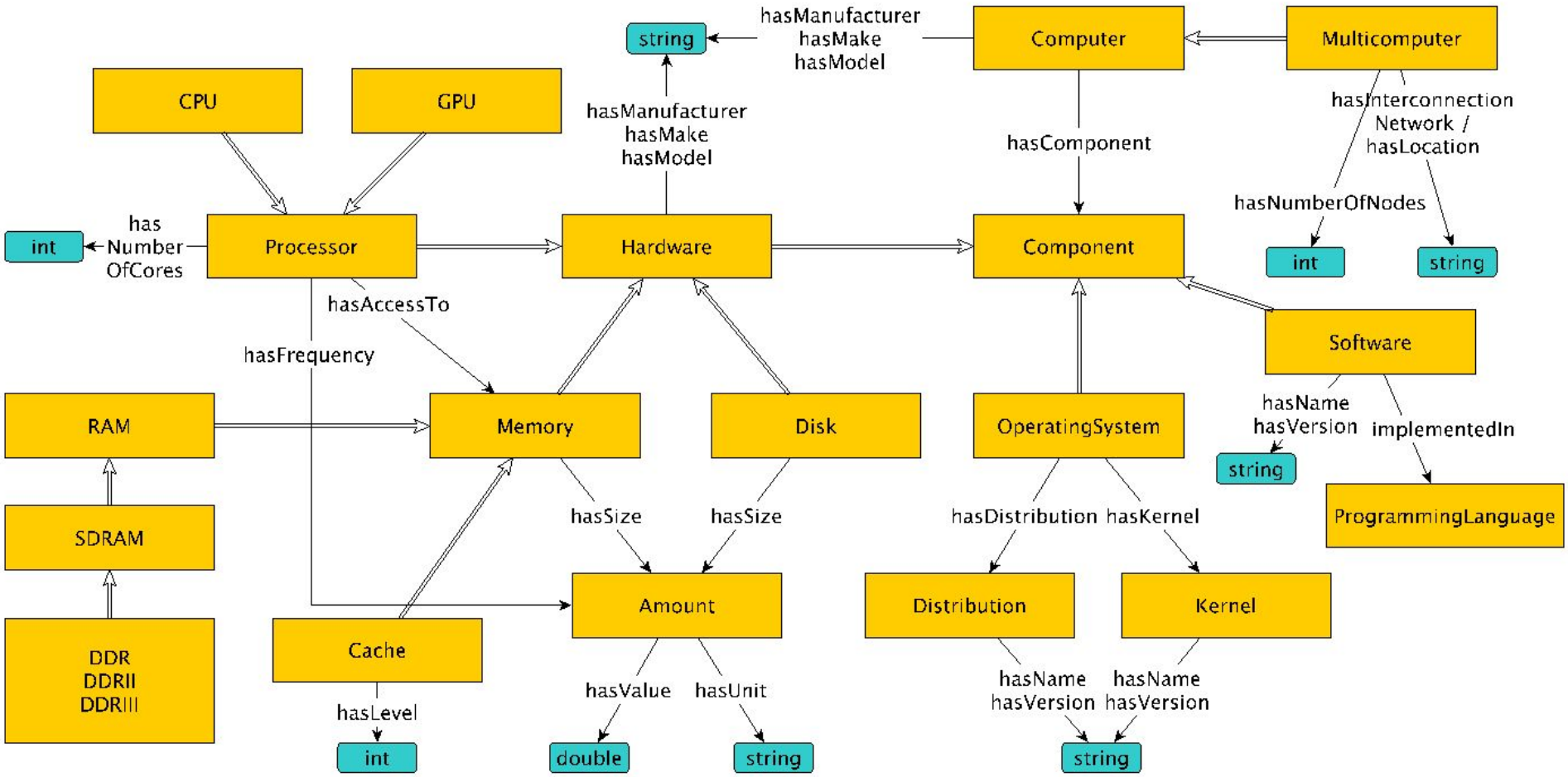
Motivation

- Discovery and reproducibility of existing results are key to scientific progress, but…
- Just finding those existing results is not enough – the **context** in which they were collected is key.

Motivation

- In previous work we created an ODP to facilitate the following competency questions:
  - What environment do I need to put in place in order to replicate the work in Paper X?
  - There has been an error found in Script Y.Which analyses need to be re-run?
  - Based on recent research in Field Z, what tools and re- sources should new students work to become familiar with?
  - Are the results from Study A and Study B comparable from a computational environment perspective?

The Computational Environment ODP



Next Step: Populating the pattern

- Our goal in this work was to populate the pattern based on descriptions of computational environments mentioned in academic papers.

The algorithm is implemented in C/C++ and run on a 16 core red hat linux box with 3.2GhZ HT Xeon CPU and 8GB RAM.

cpu make: HT Xeon
cpu frequency value: 3.2
cpu frequency unit: GHz
cpu num−cores: 16
memory type: RAM
memory size value: 8
memory size unit: GB
programming−language: C/C++
os kernel name: Linux
os distribution name: Red Hat

Problem Formulation

- We formulate this problem solely as a Named Entity Recognition (NER) task.
- In other words, we're trying to directly arrive at the tags shown on the previous slide, with the intention of later creating SPARQL construct queries to expand these tags into the terminology used by the ODP.
- If more than one computational environment is described in an article, the appropriate relations between instances can be then determined based on proximity in the underlying text.
- This is in contrast to many other approaches that divide this process into two steps: entity recognition/typing and relation extraction.

Approach 1: Machine Learning

- Conditional Random Field classifier (Stanford NLP group's implementation)
- Uses features such as word order, n-grams, part of speech, and word shape to create a probabilistic model to predict the tags in previously unseen documents

```
running   O
on        O
a         O
PC        O
with      O
Linux     os_kernel_name
CentOS    os_distribution_name
5         os_distribution_version
,         O
1         memory_size_value
GB        memory_size_unit
memory    O
and       O
2.4       cpu_frequency_value
GHz       cpu_frequency_unit
CPU       O
```

Approach 2: Rules

- Token-based regular expressions
- Can incorporate text, part-of-speech, and previously assigned tags
- The rule below states that any noun phrase that is between an operating system kernel name and the word “version” should be tagged as an operating system distribution name

```
ruleType: “tokens”
pattern: ([{ner:os_kernel_name}]
          ([{pos:NN} | {pos:NNP}) &
          !{word:/(?i)version/}])),
action: (Annotate ($1, ner, “os_distribution_name”)),
stage: 2
```

Dataset

- 100 academic articles published in 2012 or later (called “current”) and 20 published in or before 2002 (called “old”)
- 20 current papers and 4 old ones from biology, chemistry, engineering, mathematics, and physics
- One-fifth of the current papers were randomly selected for the training set; remainder is the test set
- Manually tagged by a single person; issues resolved by consulting manufacturer’s websites, websites about historical computing technologies, and really geeky people

The Challenge

- Computer technology is a fast-changing field, so NER models trained on older documents may degrade in performance over time.

- Some examples:
  - The most common programming language mentioned in older documents is FORTRAN
  - DEC used to be a commonly mentioned computer manufacturer
  - Old documents seldom contain any information at all about GPUs

Methodology

- We trained both the CRF and rules-based classifiers on the old documents and 20% of the new documents
- The classifiers were tested on the remaining 80% of the new documents
- Evaluation was in terms of precision, recall and F-measure

Results

Machine Learning

Dataset	Model	Precision	Recall	F-measure
Training	Current Old	0.95 0.95	0.93 0.93	0.94 0.94
Test	Current Old	0.90 1.00	0.52 0.01	0.66 0.02

Rules

Dataset	Model	Precision	Recall	F-measure
Training	Current Old	0.96 0.99	0.93 0.94	0.95 0.97
Test	Current Old	0.79 0.80	0.67 0.28	0.73 0.41

Analysis

- Machine Learning
  - The performance of both the old and current model is very good on the training data
  - Performance of the current model on the test set is reasonable
  - Performance of the old model on the test set is abysmal

- Rules
  - Same pattern as above, but…
  - Performance on the test set is better in both cases
    - Current model: .73 vs .66
    - Old model: .41 vs .02

Can we do better?

- The rules-based classifier was clearly better than the machine learning approach in this instance, but performance still degraded significantly when the older model was used on newer documents

- Developing new rule sets is a very time-consuming task (each rules-based model used here took two days to create)

Key Insight

- An underlying problem is that some tags' rules are based on other tags. Missing one causes the system to miss both.

Lenovo	Thinkpad		Lenovo	Thinkpad
computer manufacturer	???	→	computer manufacturer	computer make

Lenovo	Thinkpad		Lenovo	Thinkpad
???	???	→	???	???

Approach 3: Rules + Background Knowledge

- Developed a custom NER module that fits into the Stanford NLP pipeline
- When tagging a document, the module queries Wikipedia for each proper noun in the key paragraph(s) and if a page is returned, determines which tag, if any, is most relevant
- Relevance is determined by counting the number of times each tag appears in the first three sentences of the document and weighting tags that appear earlier more than those that appear later.
- After the Wikipedia annotator finishes, the rules-based annotator is run.

Results

Rules

Dataset	Model	Precision	Recall	F-measure
Training	Current	0.96	0.93	0.95
	Old	0.99	0.94	0.97
Test	Current	0.79	0.67	0.73
	Old	0.80	0.28	0.41

Rules + Background Knowledge

Dataset	Model	Precision	Recall	F-measure
Training	Current	0.93	0.93	0.93
	Old	0.93	0.93	0.93
Test	Current	0.71	0.70	0.70
	Old	0.61	0.41	0.49

Results

- Using the background knowledge improves performance of the old model by 20% (0.49 versus 0.41)
- Performance of the current model is reduced by 4 percent (0.70 versus 0.73 F-measure)
- We conclude that this approach should not be used unless it is warranted by changes in the domain of interest and the age of the model used for ontology population.

Questions?