

Automatic Extraction of a WordNet-like Identifier Network from Software

ICPC 2010

Jean-Rémy Falleri, M. Huchard, M. Lafourcade, C. Nebut, V. Prince,
M. Dao

INRIA Lille Nord Europe

30/06/2010

Program identifiers

- ▶ Name given by the developers to the elements of a program
 - ▶ *LinkedList*, *getNextWarning*
- ▶ They represent about 33% of the tokens of a program
- ▶ They include important insight regarding the program semantic
- ▶ Identifiers, like words, entertain Lexical relations
 - ▶ *getNextWarning*, *getPreviousWarning*

Lexical relations

- ▶ Lexical relations are useful to navigate and reason in a set of words
 - ▶ WordNet
- ▶ The most common relations are hypernymy, hyponymy and synonymy
- ▶ Retrieving these relations between program identifiers could ease several tasks
 - ▶ Program navigation, program labeling, naming assistance

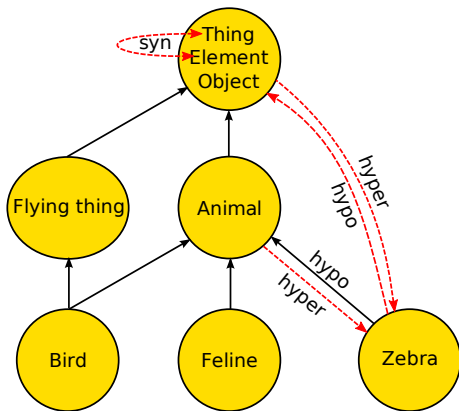
Extracting lexical relations

- ▶ Manually
 - ▶ Lot of work
- ▶ Using pattern matching
 - ▶ Not applicable on identifiers

Our work

**Extract automatically these relations in a wordnet like structure,
called lexical view**

Lexical view



Extraction process

Five step process

1. Tokenization
2. Part-of-speech (POS) *tagging*
3. Dependency sorting
4. Lexical expansion
5. Lexical view computation

An identifier set

- ▶ *getNextWarning*
- ▶ *set_Warning*
- ▶ *getPreviousWarning*

Tokenization

Description

- ▶ Split the identifiers into list of tokens
- ▶ We use common syntactical clues
 - ▶ Case changes
 - ▶ Non-alphabetic characters

Before

- ▶ *getNextWarning*
- ▶ *set_Warning*
- ▶ *getPreviousWarning*

Tokenization

Description

- ▶ Split the identifiers into list of tokens
- ▶ We use common syntactical clues
 - ▶ Case changes
 - ▶ Non-alphabetic characters

After

- ▶ *(get, Next, Warning)*
- ▶ *(set, Warning)*
- ▶ *(get, Previous, Warning)*

Part-of-speech (POS) tagging

Description

- ▶ Compute a POS for each term of each token list
- ▶ We use the *Tree Tagger* tool

Before

- ▶ *(get, Next, Warning)*
- ▶ *(set, Warning)*
- ▶ *(get, Previous, Warning,)*

Part-of-speech (POS) tagging

Description

- ▶ Compute a POS for each term of each token list
- ▶ We use the *Tree Tagger* tool

After

- ▶ ((*get*, *Verb*), (*Next*, *Adj*), (*Warning*, *Noun*))
- ▶ ((*set*, *Verb*), (*Warning*, *Noun*))
- ▶ ((*get*, *Verb*), (*Previous*, *Adj*), (*Warning*, *Noun*))

Dependency sorting

Description

- ▶ Sort the tokens of a list using the domination order
- ▶ Use of heuristics

Before

- ▶ $((get, Verb), (Next, Adj), (Warning, Noun))$
- ▶ $((set, Verb), (Warning, Noun))$
- ▶ $((get, Verb), (Previous, Adj), (Warning, Noun))$

Dependency sorting

Description

- ▶ Sort the tokens of a list using the domination order
- ▶ Use of heuristics

After

- ▶ *((get, Verb), (Warning, Noun), (Next, Adj))*
- ▶ *((set, Verb), (Warning, Noun))*
- ▶ *((get, Verb), (Warning, Noun), (Previous, Adj))*

Lexical expansion

Description

- ▶ Addition of implicit list of tokens
- ▶ Use of longest common prefix between the sorted token lists

Before

- ▶ $((get, Verb), (Warning, Noun), (Next, Adj))$
- ▶ $((set, Verb), (Warning, Noun))$
- ▶ $((get, Verb), (Warning, Noun), (Previous, Adj))$

Lexical expansion

Description

- ▶ Addition of implicit list of tokens
- ▶ Use of longest common prefix between the sorted token lists

After

- ▶ *((get, Verb), (Warning, Noun))*
- ▶ *((get, Verb), (Warning, Noun), (Next, Adj))*
- ▶ *((set, Verb), (Warning, Noun))*
- ▶ *((get, Verb), (Warning, Noun), (Previous, Adj))*

Lexical view computation

Description

- ▶ Order: *is prefix of* between the token lists
- ▶ We compute the transitive reduction

Avant

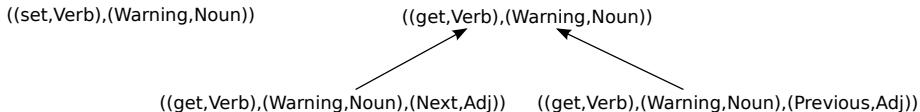
- ▶ $((get, Verb), (Warning, Noun))$
- ▶ $((get, Verb), (Warning, Noun), (Next, Adj))$
- ▶ $((set, Verb), (Warning, Noun))$
- ▶ $((get, Verb), (Warning, Noun), (Previous, Adj))$

Lexical view computation

Description

- ▶ Order: *is prefix of* between the token lists
- ▶ We compute the transitive reduction

Après

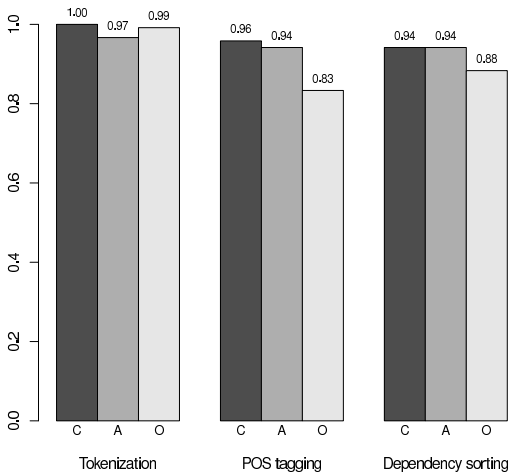


Evaluation of the extraction process

- ▶ Corpus of 360 identifiers (classes, attributes and methods) manually treated
 - ▶ Drawn at random in 24 Java programs
 - ▶ 120 class identifiers, 120 attribute identifiers and 120 method identifiers
- ▶ Validation of the steps *Tokenization*, *POS tagging* et *Dependency sorting*

Results

Accuracy of the NLP techniques



Evaluation of the produced lexical views

- ▶ Corpus of 24 Java programs
- ▶ For each program, we build a lexical view for:
 - ▶ The class identifiers
 - ▶ The attributes identifiers
 - ▶ The method identifiers

Results

