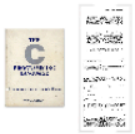
Comments and Code as Bitext

James Paul White
jimwhite@uw.edu
Department of Linguistics (CLMS)
University of Washington

A Long and Winding Road

Learning C as a Second Language



Frames and Lexical Semantics in the Computer Domain

Computer Language as Human Language

The Linguistic Interpretation of Computer Programs (and Related Text)

Jim White
February 3, 2012

A CLMS Thesis Proposal
Under Construction

Apply the Methods of Computational Linguistics to the Problems of Computer Program Understanding

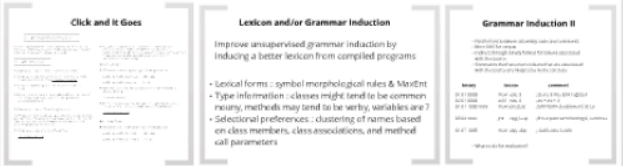
↓

Acquire Natural Language and World Knowledge by Reading Computer Programs and/or Related Text

Computer Programs and Related Text

A Brief Personal History and Some Philosophy

Choosing a Problem and an Approach



EMNLP 2013
Twenty Years of Bitext Workshop

- **1993** a landmark year in empirical methods for processing parallel corpora
- William A. Gale and Kenneth Ward Church. **A program for aligning sentences in bilingual corpora.** *Computational Linguistics*, 19(1), 1993.
  - Included C source code for the program *align*.
- Peter F. Brown, Stephen A. Della-Pietra, Vincent J. Della-Pietra, and Robert L. Mercer**. The mathematics of statistical machine translation.** *Computational Linguistics*, 19(2), 1993.
  - IBM word alignment models 1 thru 5
- **First Workshop on Very Large Corpora**
  - http://aclweb.org/anthology/sigdat.html#1993_0
  - Led to first conference on Empirical Methods in Natural Language Processing in 1996

What is Bitext?

- Jörg Tiedemann. **Bitext Alignment**. *Synthesis Lectures on Human Language Technologies,* 4(2), 2011.
- Originally used to refer to **documents and their translations into other languages** for use in translation studies
  - Brian Harris. Bi-Text, a new concept in translation theory. *Language Monthly*, 54, 1988.
- Now commonly used to refer to a broader range of parallel resources including multiple translations in same language
- Related Terms
  - Parallel Text — same as bitext in CL, comparable in TS
  - Comparable Text — same domain, differing languages
  - Parallel Corpora — collections of bitext
  - Translation Corpora — stricter form of parallel corpora

Bitext in the Wild

- Fung, Pascale, and Percy Cheung. "**Multi-level bootstrapping for extracting parallel sentences from a quasi-comparable corpus**." *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, 2004.
  - A "completely unsupervised" method for extracting parallel sentences from **quasi-comparable** corpora.
  - A **noisy parallel** corpus has documents which contain many parallel **sentences** in **roughly the same order**.
  - **Comparable** corpora contain **topic-aligned** documents which are **not translations** of each other.
  - **Quasi-comparable** corpora contain **bilingual** documents which are **not necessarily on the same topic**.
- Smith, Jason R., Chris Quirk, and Kristina Toutanova. "**Extracting parallel sentences from comparable corpora using document level alignment**." *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010.
  - Wikipedia is a "surprisingly" useful source of parallel data.

Poster Pitch : Two Minutes Max

- Computer programs and related text are a vast untapped source of bitext
  - Billions of lines of Open Source Software and growing exponentially since 1995 (and until ?)
- Collocated with the human language labels, strings, and comments in source files, programmers create elaborate semantic annotations in computer language
- There are then document-level relationships between source files, data files, and documentation files
- Bitext in which one of the languages is a computer language is very useful because there are many things a computer can do with language it can parse reliably and evaluate

Poster Pitch : Two Minutes Max

- As with other noisy parallel, comparable, and quasi-comparable corpora there are many and varied forms of translational relationships between the texts
- The Pedantic Javadoc Corpus is a small bitext corpus (currently 100 pairs) of English comments and simple Java statements
- Created to begin exploring the semantic representation issues and as a test for automatic classification of such pairs
- One application I envision would use SMT methods to generate code for trials within a reinforcement learning framework for following instructions such as README files

Three Level Models for Conceptual Representation

Popular Models in Psychology/Cognitive Science/Artificial Intelligence

Roger Schank & Robert Abelson	David Marr	Zenon Pylyshyn	Glass, Holyoak, and Santa
Goal	**Computational**	Semantic	Content
Plan	**Algorithmic**	Syntactic	Form
Action	**Implementational**	Physical	Medium

Schank, Roger C., and Robert P. Abelson. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. Psychology Press. (1977).

Marr, David. *Vision: A Computational Approach*. (1982).

Gentner, Dedre. "Psychology in cognitive science: 1978–2038." *Topics in Cognitive Science* 2.3 (2010): 328-344.

Levels of Representation for Code

- Java
  - ```
    void bubblesort(int[] s) {
        …
    }
    ```
- Goal/Computational
  - Sort the given list of numbers.
  - Upon return, the list must be in sorted order with the smallest element first.
  - This method sorts a list of integers into increasing order using Bubblesort.
- Plan/Algorithmic
  - Scan the list from beginning to end, swapping pairs where the first is larger than the second. Repeat until no swaps are performed.
- Action/Implementational
  - Set the swapped flag. While swapped is true, reset the flag then scan for pairs needing a swap. To scan for pairs needing a swap, with an index starting from zero and incrementing until it is the length of the array minus one, compare the numbers at the index and index plus one. If the first number is larger than the second, exchange them and set the swapped flag to true.
    ```

A Simple Example

Get the "<B>" string.

com.sun.tools.doclets.formats.html.markup.HtmlWriter

```
public String getBold() {
    return "<B>";
}
```

Return, text passed, with Italics <I> and </I> tags, surrounding it.

```
public String italicsText(String text) {
    return "<I>" + text + "</I>";
}
```

Not Pedantic

Start a set of fresh registers.

```
com.sun.tools.javac.jvm.Code
void newRegSegment() {
    nextreg = max_locals;
}
```

```
(Vars
    (((IDENTIFIER 0)(Name max_locals))
    ((IDENTIFIER 1)(Name nextreg))))

(Tree (EXPRESSION_STATEMENT
    (Expression (ASSIGNMENT
        (Expression (IDENTIFIER 0))
        (Variable (IDENTIFIER 1))))))
```

Non-Pedantic Examples

Is c a printable ASCII character?

```
com.sun.tools.apt.mirror.declaration.Constants.Formatter
static boolean isPrintableAscii(char c) {
    return c >= ' ' && c <= '~';
}
```

- The Javadoc is a translation of the programmer's phrase for the method's name and therefore literal or pedantic in that sense
  - See the Java Programmer's Phrase Book slide here later
- The Javadoc's relationship to the method body though is mostly at goal level
  - The top-level is actually pedantic since "is" maps to "return" here

Interesting Examples

Increase left margin by indentation width.

```
com.sun.tools.javac.tree.Pretty
void indent() {
    lmargin = lmargin + width;
}
```

Decrease left margin by indentation width.

```
com.sun.tools.javac.tree.Pretty
void undent() {
    lmargin = lmargin – width;
}
```

A Not So Simple Example

***Return code byte at position pc* as an unsigned int.**

```
com.sun.tools.javac.jvm.Code
int get1(int pc)
{
   return code[pc] & 255;
}

(RETURN
    (Expression (AND
        (LeftOperand (ARRAY_ACCESS
            (Expression (IDENTIFIER (Name code)))
            (Index (IDENTIFIER (Name pc)))))
        (RightOperand (INT_LITERAL (Value 255))))))
```

Divide And Conquer

Return code byte at position pc as an unsigned int.

```
(RETURN(Expression(AND
        (LeftOperand(ARRAY_ACCESS
            (Expression (IDENTIFIER (Name code)))
            (Index (IDENTIFIER (Name pc)))))
        (RightOperand(INT_LITERAL(Value 255))))))
```

```
(Vars (((IDENTIFIER 0)(Name code)) ((IDENTIFIER 1)(Name pc))))
(Tree (RETURN(Expression(AND
        (LeftOperand (ARRAY_ACCESS
           (Expression (IDENTIFIER 0))
           (Index (IDENTIFIER 1))))
        (RightOperand (INT_LITERAL(Value 255)))))))
```

Composing a Goal

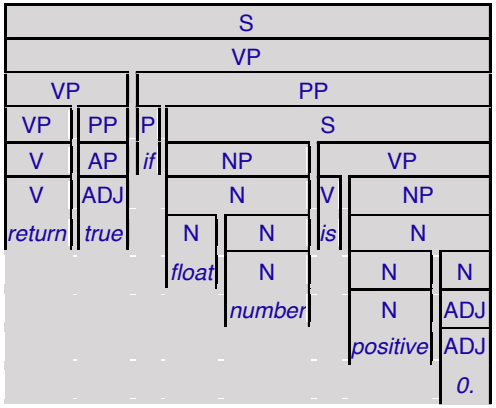
Return true iff float number is positive 0.

```
com.sun.tools.javac.jvm.Items.ImmediateItem
boolean isPosZero(x)
float x
{
    return x == 0.0F && 1.0F / x > 0.0F;
}
(Vars (((IDENTIFIER 0)(Name x))((IDENTIFIER 1)(Name x))))
(Tree (RETURN
    (Expression (CONDITIONAL_AND
        (LeftOperand (EQUAL_TO
            (LeftOperand (IDENTIFIER 0))
            (RightOperand (FLOAT_LITERAL(Value 0.0)))))
        (RightOperand (GREATER_THAN
            (LeftOperand (DIVIDE
                (LeftOperand (FLOAT_LITERAL (Value 1.0)))
                (RightOperand (IDENTIFIER 1))))
            (RightOperand (FLOAT_LITERAL (Value 0.0)))))))))
```

Bind Them Variables

Return true iff float number is positive 0.

return x == 0.0F && 1.0F / x > 0.0F;



TOP h1

INDEX e3

RELS {

pronoun_q⟨0:42⟩ [LBL h4, ARG0 x6, RSTR h5, BODY h7]

pron⟨0:42⟩ [LBL h8, ARG0 x6]

_return_v_1⟨0:6⟩ [LBL h9, ARG0 e3, ARG1 x6]

subord⟨7:11⟩ [LBL h10, ARG0 e13, ARG1 h12, ARG2 h11]

_true_a_of⟨7:11⟩ [LBL h14, ARG0 e16, ARG1 i15, ARG2 i17]

_if_x_then⟨12:14⟩ [LBL h2, ARG0 e18, ARG1 h19, ARG2 h20]

udef_q⟨15:27⟩ [LBL h21, ARG0 x23, RSTR h22, BODY h24]

compound⟨15:27⟩ [LBL h25, ARG0 e27, ARG1 x23, ARG2 x26]

udef_q⟨15:20⟩ [LBL h28, ARG0 x26, RSTR h29, BODY h30]

_float_n_1⟨15:20⟩ [LBL h31, ARG0 x26]

_number_n_of⟨21:27⟩ [LBL h25, ARG0 x23]

_be_v_id⟨28:30⟩ [LBL h32, ARG0 e33, ARG1 x23, ARG2 x34]

number_q⟨31:42⟩ [LBL h35, ARG0 x34, RSTR h36, BODY h37]

compound⟨31:42⟩ [LBL h38, ARG0 e40, ARG1 x34, ARG2 x39]

udef_q⟨31:39⟩ [LBL h41, ARG0 x39, RSTR h42, BODY h43]

_positive_n_1⟨31:39⟩ [LBL h44, ARG0 x39]

basic_card⟨40:42⟩ [LBL h38, ARG0 x34, ARG1 i45, CARG *TOP*]

}

HCONS { h1 =q h2, h5 =q h8, h12 =q h9, h11 =q h14, h19 =q h10, h20 =q h32, h22 =q h25, h29 =q h31, h36 =q h38, h42 =q h44 }

Related Work

- All sorts of Computer Science and Software Engineering Research
  - Static & Dynamic Code Analysis
  - Formal Verification & Proof Methods
  - Requirements and Specifications Traceability
  - Program Understanding & Program Comprehension Tools
    - Currently directed at human programmers and effectiveness is typically evaluated using ethnographic and CogSci experiments
- Open Ontology / Folksonomy Reasoning
  - WordNet, Freebase, etc
  - The Semantic Web

The Java Programmer's Phrase Book

accept, action, add, check, clear, close, create, do, dump, end, equals, find, get, has, init, initialize, insert, is, new, next, parse, print, process, read, remove, reset, run, set, size, start, update, validate, visit

Fig. 4. The **is-*** branch of phrases

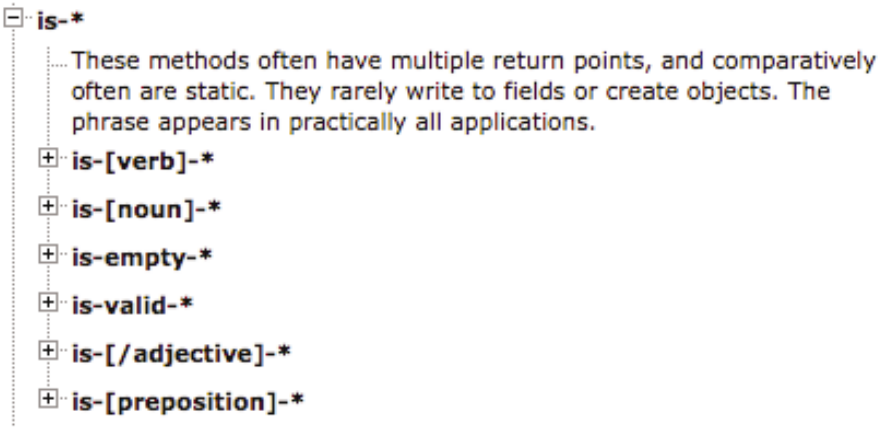


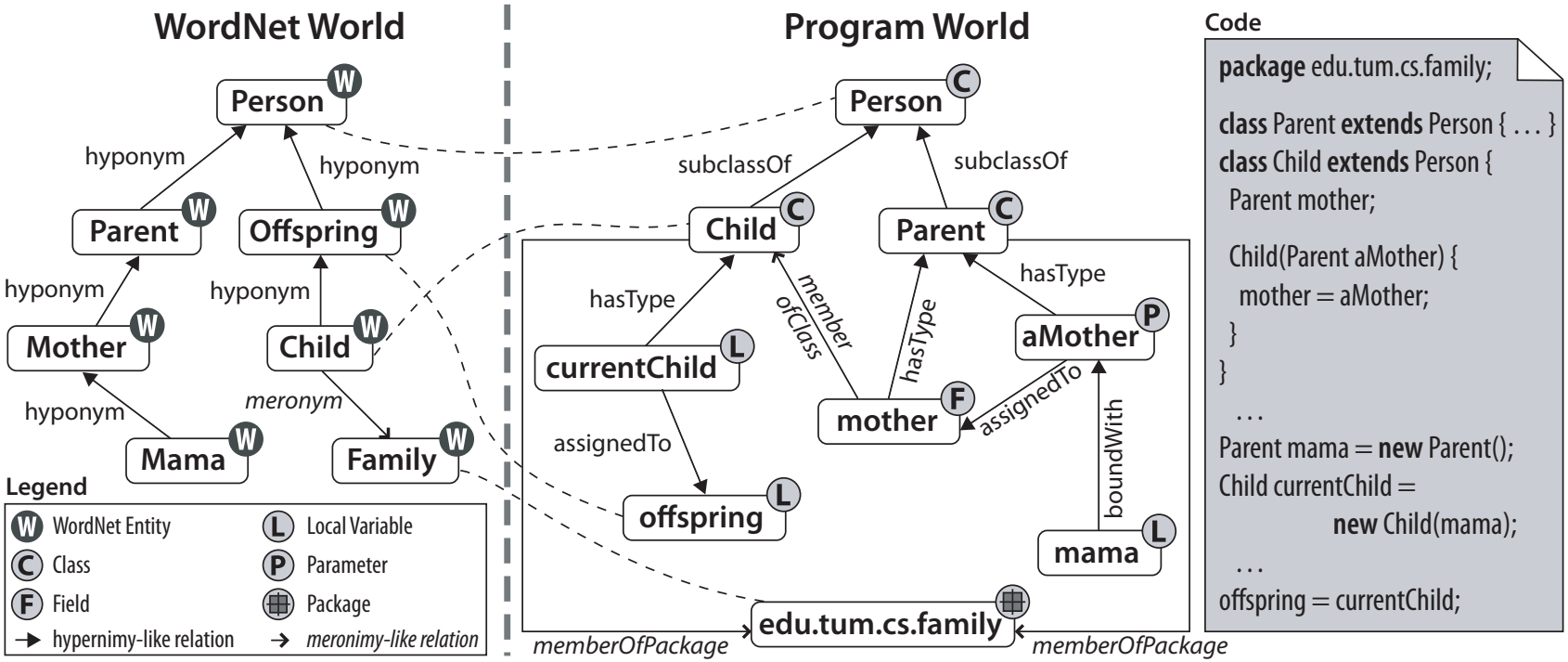
Table 5. Distribution of grammatical structures

Structure	*Instances*	*Percent*
[verb]-[noun+]	422546	39.45%
[verb]	162050	15.13%
[verb]-[type]	78632	7.34%
[verb]-[adjective]-[noun+]	74277	6.93%
[verb]-[adjective]	28397	2.65%
[noun+]	26592	2.48%
[verb]-[noun+]-[type]	18118	1.69%
[adjective]-[noun+]	15907	1.48%
[noun+]-[verb]	14435	1.34%
[preposition]-[type]	13639	1.27%

Einar W Høst & Bjarte M Østvold. (2009). The Java Programmer's Phrase Book. Presented at the First International Conference of Software Language Engineering, Berlin, pp. 322–341, 2009.

Concept Assignment

Ted J Biggerstaff, Bharat G Mitbander, & Dallas E Webster. Program understanding and the concept assignment problem. *Communications of the ACM*, *37*(5), 1994.



Daniel Ratiu and Florian Deissenboeck. Programs are Knowledge Bases. *ICPC 2006, 14th IEEE International Conference on Program Comprehension*. 2006.

Future Work

- Linux Package Corpus
  - Package metadata curated by package maintainers
  - Includes download link, package dependency expressions, single line and single paragraph description summaries, unpack, build, and install scripts
  - Currently RPM, easily extend to Debian and Nix
- Click & It Goes
  - Apply Reinforcement Learning to generate build and install scripts from the README and INSTALL files
  - Test by comparing whether scripts generated for packages with dependencies result in their dependents still working

Thank You!

James Paul White
jimwhite@uw.edu
Department of Linguistics (CLMS)
University of Washington