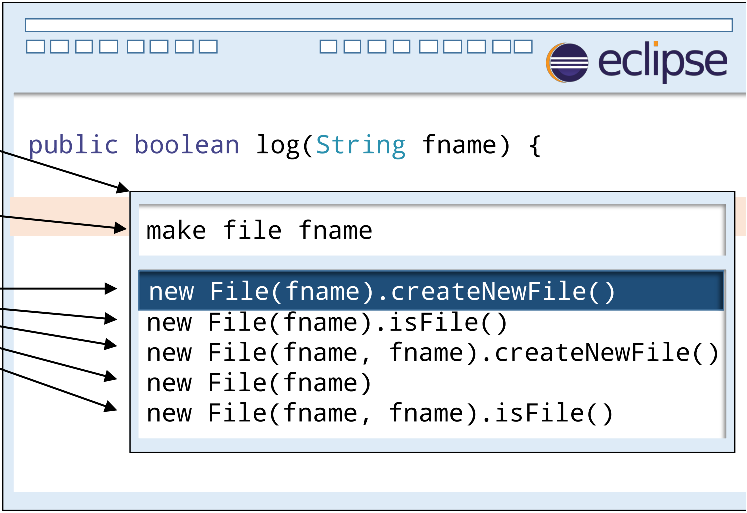
Synthesizing **Java** expressions from free-form **queries**

completion box

query
English + identifiers

Java Expressions
- query relevant
- type & scope correct
- statistically likely,
  yet not copy-pasted



```
public boolean log(String fname) {
```

```
make file fname
```

```
new File(fname).createNewFile()
new File(fname).isFile()
new File(fname, fname).createNewFile()
new File(fname)
new File(fname, fname).isFile()
```

Examples of results that anyCode gives

load class ”MyClass.class”

```
Thread.currentThread()
      .getContextClassLoader()
      .loadClass(”MyClass.class”)
```

write ”hello” to file ”text.txt”

```
FileUtils.writeStringToFile(
     new File(”text.txt”), ”hello”)
```

new buffered stream ”text.txt”

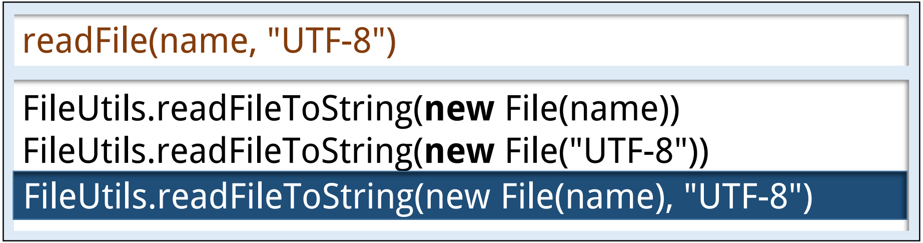
```
new BufferedReader(
  new InputStreamReader(
  new BufferedInputStream(
  new FileInputStream(”text.txt”))))
```

set thread max priority

```
Thread.currentThread()
  .setPriority(Thread.MAX_PRIORITY)
```

Can also help correct “sloppy Java”

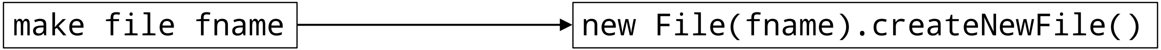
```
public String prepareMessage(String name, String protocol)
                                                    throws Exception {
    if (!protocol.equals("file"))
        return errorMessage(protocol);
    else
        return
}
```



```
readFile(name, "UTF-8")
```

```
FileUtils.readFileToString(new File(name))
FileUtils.readFileToString(new File("UTF-8"))
FileUtils.readFileToString(new File(name), "UTF-8")
```

Translation problem



`make file fname` → `new File(fname).createNewFile()`

English queries:

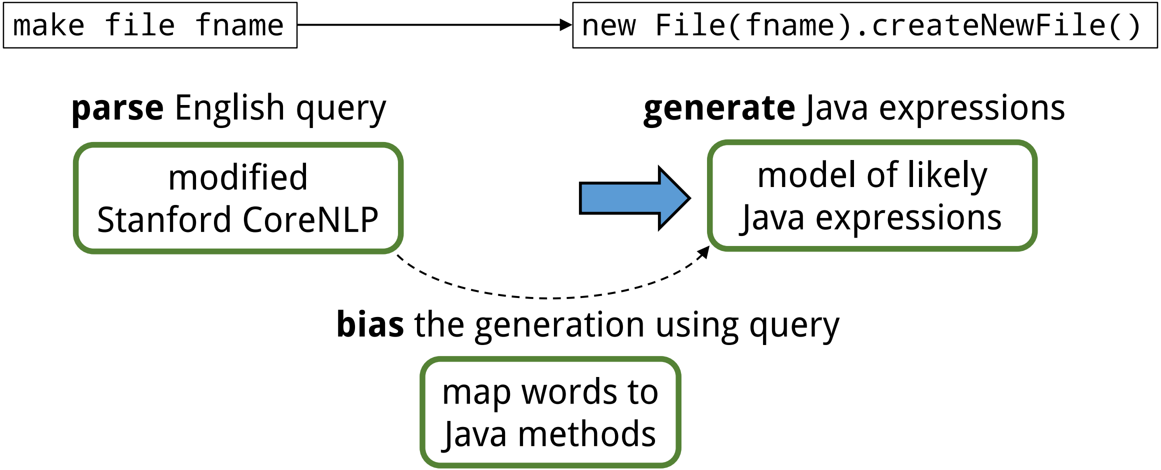
- English phrase structures
- English dictionary words
- identifiers in scope
- literals, e.g. 42 or “Hello”

Java expressions:

- scoping and type rules of Java
- API method names `camelCase`
- identifiers in scope
- literals, e.g. 42 or “Hello”

No readily available large-scale parallel corpus, unlike machine translation.

Key tasks in translation



`make file fname` → `new File(fname).createNewFile()`

parse English query

modified Stanford CoreNLP

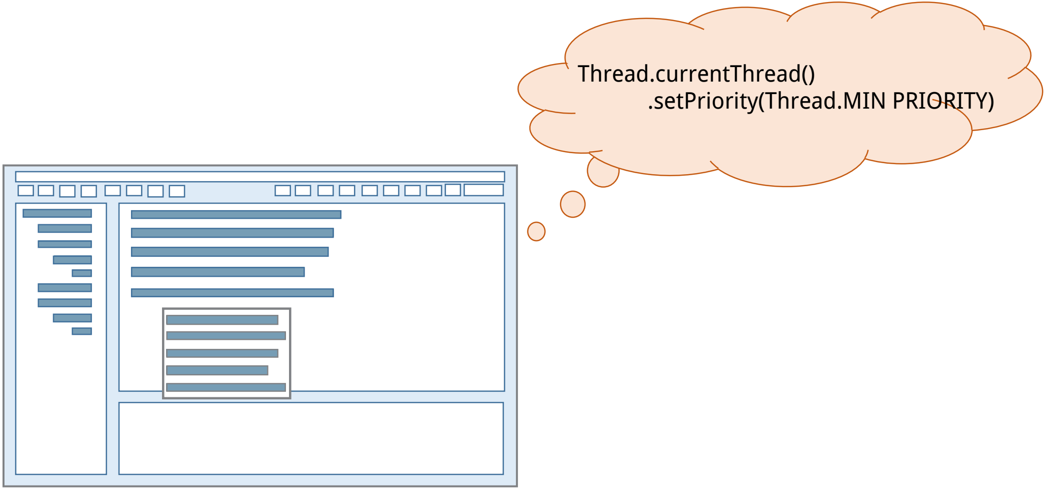
generate Java expressions

model of likely Java expressions

bias the generation using query

map words to Java methods

Which Java expressions do IDEs dream about?

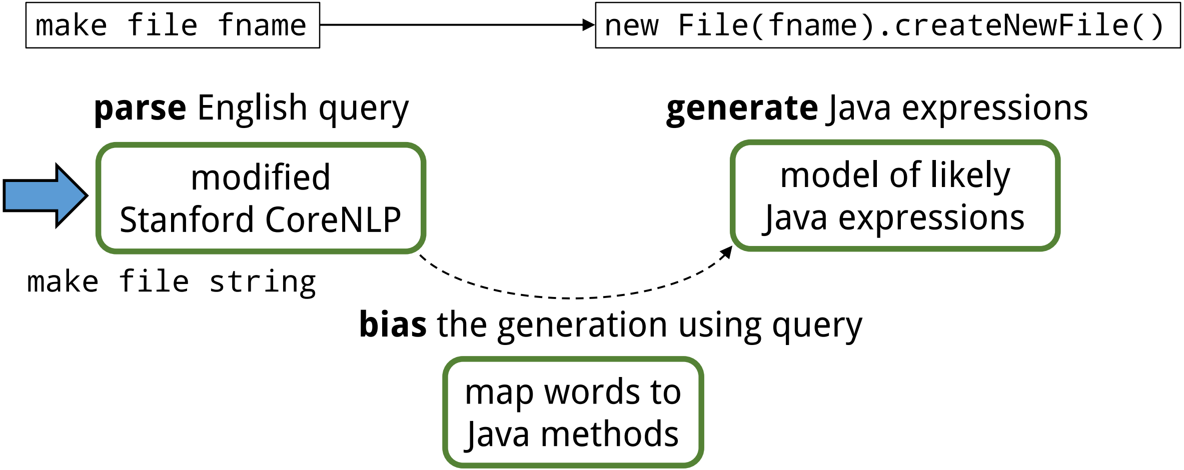


```
Thread.currentThread()
        .setPriority(Thread.MIN PRIORITY)
```

Distribution over all Java expressions

- Our prior work: declaration frequencies only (Gvero et al. PLDI'13)
- This work: computes additionally probabilistic context-free grammar (**PCFG**) describing likely composition of declarations
  - parse and type check 14'000 Java projects (~2M files)
  - extract PCFG from expressions, built after copy propagation on the files
  - splits Java types according to methods that return them
- Pr(expression) = product of Pr of rules used to build it
- Our model can be used for various program synthesis tasks
  - avoids bizarre solutions for highly underspecified queries
- Here: it gives baseline expression probability, in absence of a query
  - machine translation terminology: model for the target language

Key tasks in translation



`make file fname` → `new File(fname).createNewFile()`

parse English query

modified
Stanford CoreNLP

`make file string`

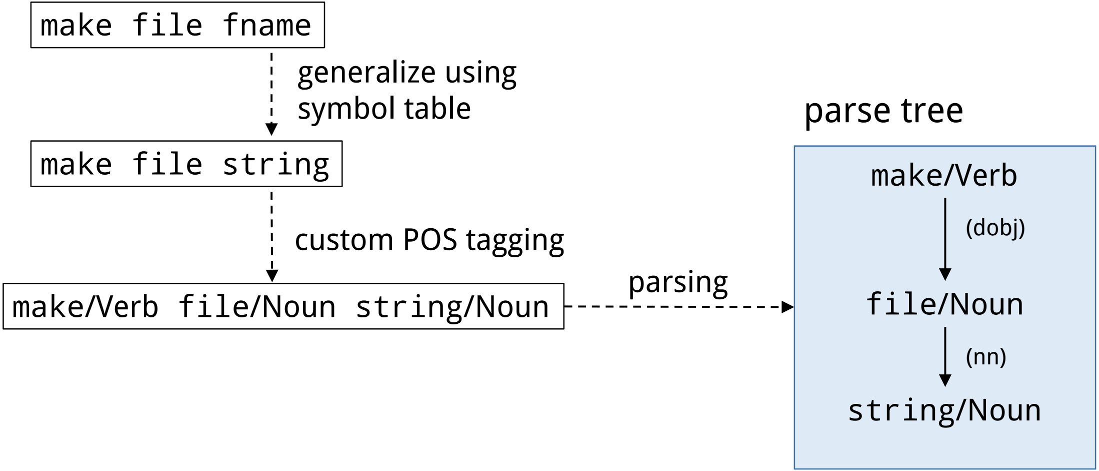
generate Java expressions

model of likely
Java expressions

bias the generation using query

map words to
Java methods

Parsing using modified CoreNLP toolkit



`make file fname`

generalize using symbol table

`make file string`

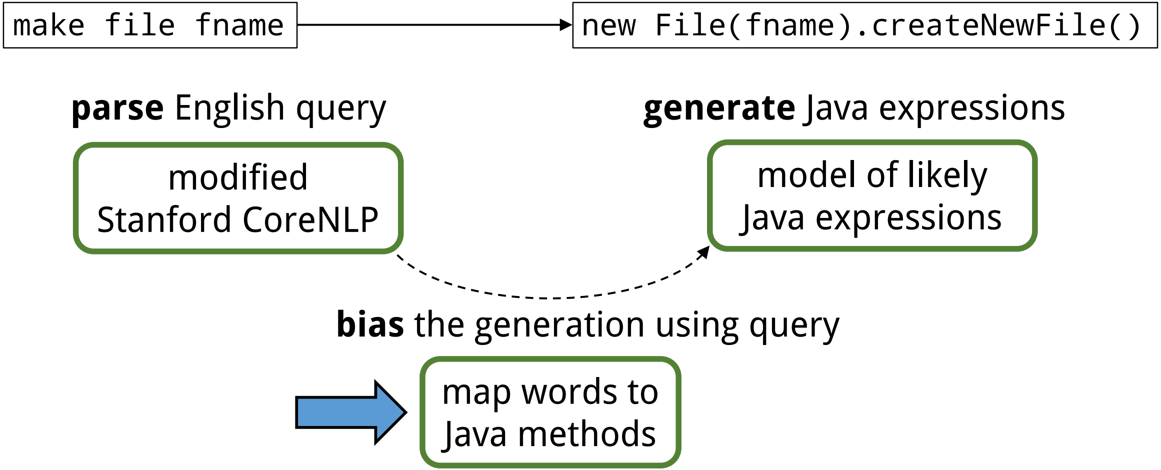
custom POS tagging

`make/Verb file/Noun string/Noun`

parsing

parse tree

Key tasks in translation



```
make file fname
```

```
new File(fname).createNewFile()
```

parse English query

modified Stanford CoreNLP

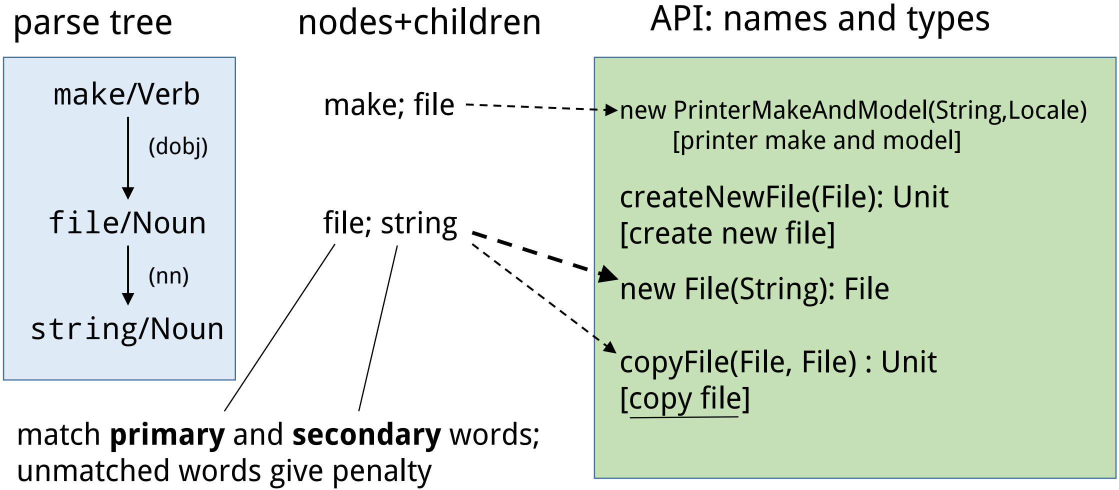
generate Java expressions

model of likely Java expressions

bias the generation using query

map words to Java methods

Map groups from parse tree to declarations



parse tree

make/Verb

(dobj)

file/Noun

(nn)

string/Noun

nodes+children

make; file

file; string

API: names and types

new PrinterMakeAndModel(String,Locale)
[printer make and model]

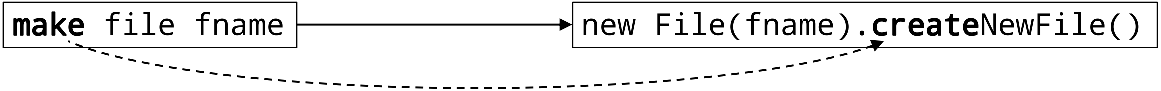
createNewFile(File): Unit
[create new file]

new File(String): File

copyFile(File, File) : Unit
[copy file]

match **primary** and **secondary** words;
unmatched words give penalty

Supporting related words

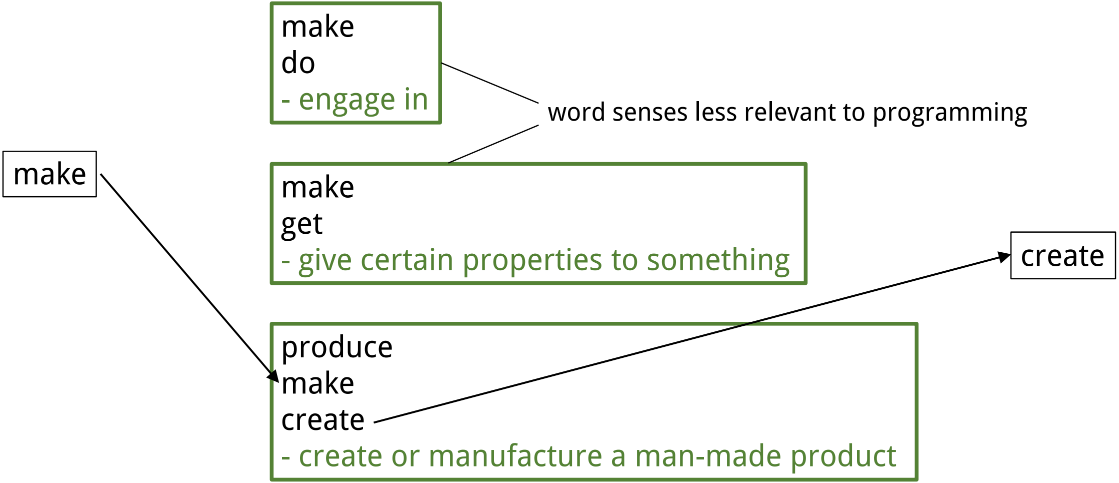


make file fname

new File(fname).**create**NewFile()

- Approach so far would not support e.g. synonyms
- We therefore use WordNet (https://wordnet.princeton.edu/)
  - Groups words into sets of synonyms (synsets)
  - Each word may belong to multiple synsets (meanings of a word)
  - Relationships between synsets, such as “is-a”
  - Synsets have English descriptions, as in a dictionary
- When computing if words are related, we favor those synsets whose description uses API words – specialize to jargon of programming

Related words through WordNet synsets



make

make
do
- engage in

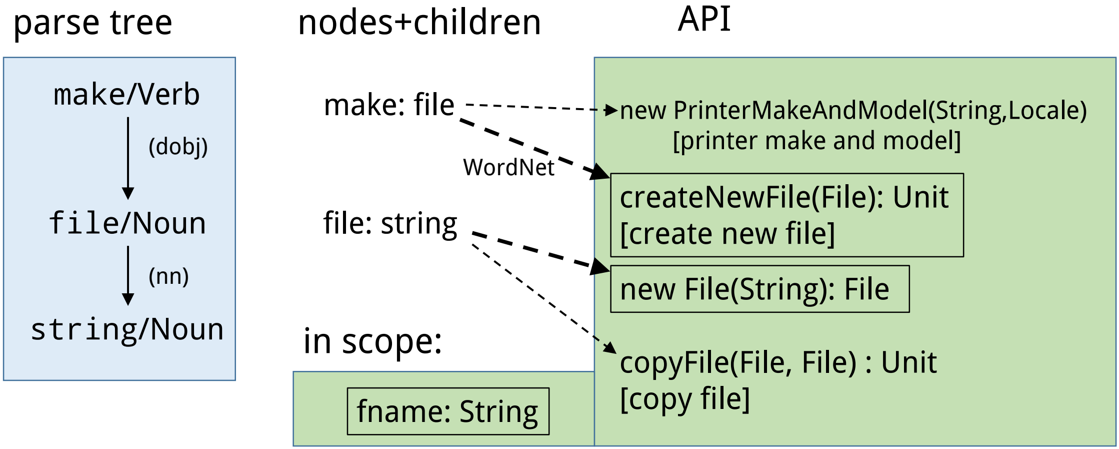
word senses less relevant to programming

make
get
- give certain properties to something

create

produce
make
create
- create or manufacture a man-made product

Map groups from parse tree to declarations



parse tree

make/Verb

(dobj)

file/Noun

(nn)

string/Noun

nodes+children

make: file

WordNet

file: string

in scope:

fname: String

API

new PrinterMakeAndModel(String,Locale)
[printer make and model]

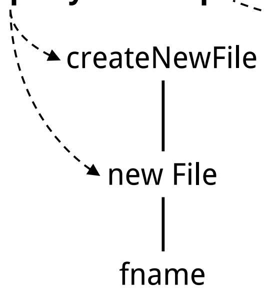
createNewFile(File): Unit
[create new file]

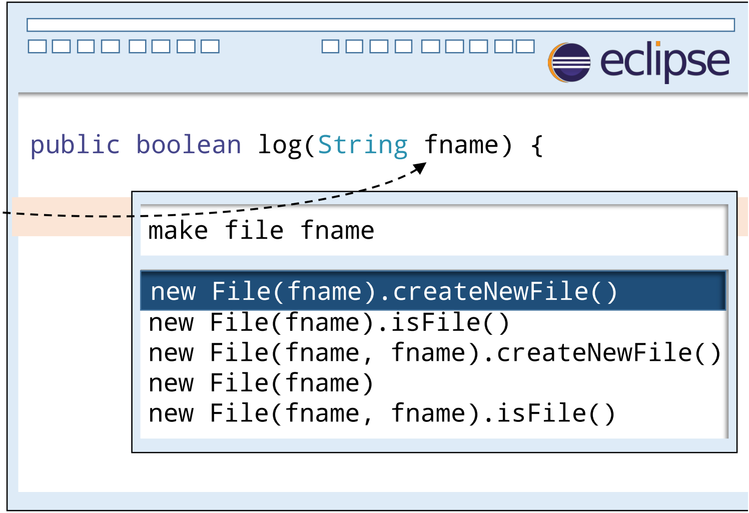
new File(String): File

copyFile(File, File) : Unit
[copy file]

Combining declarations into expression

Find most likely word from a new PCFG:
PCFG for Java
extended with bias from:
query and **scope**





```
public boolean log(String fname) {
```

```
make file fname
```

```
new File(fname).createNewFile()
new File(fname).isFile()
new File(fname, fname).createNewFile()
new File(fname)
new File(fname, fname).isFile()
```

Parameters and tuning

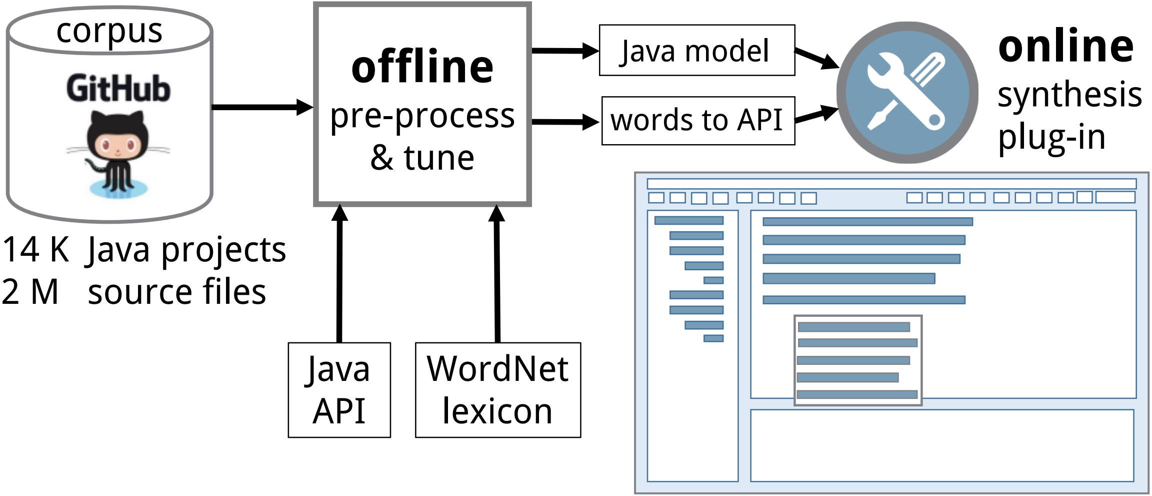
Parameters determine relative strength of different criteria

- matching of words to declarations: primary vs secondary words
- weights derived from corpus vs identifiers in scope
- order of parameters in input vs output – penalize inversion
- repetition of input elements undesired

A small number of parameters, <10

- system works even with our “best guess” values of parameters
- tuning: make it work better, by finding locally optimal values
- use local search, cost function as black box (discretize space)

Outline of our system



corpus

GitHub

14 K Java projects
2 M source files

offline
pre-process
& tune

Java model

words to API

Java API

WordNet lexicon

online
synthesis
plug-in