Data-Mining for Software Engineering: Nearest Neighbors

Martin Monperrus

version of Dec 4, 2012

Sources

Content from:

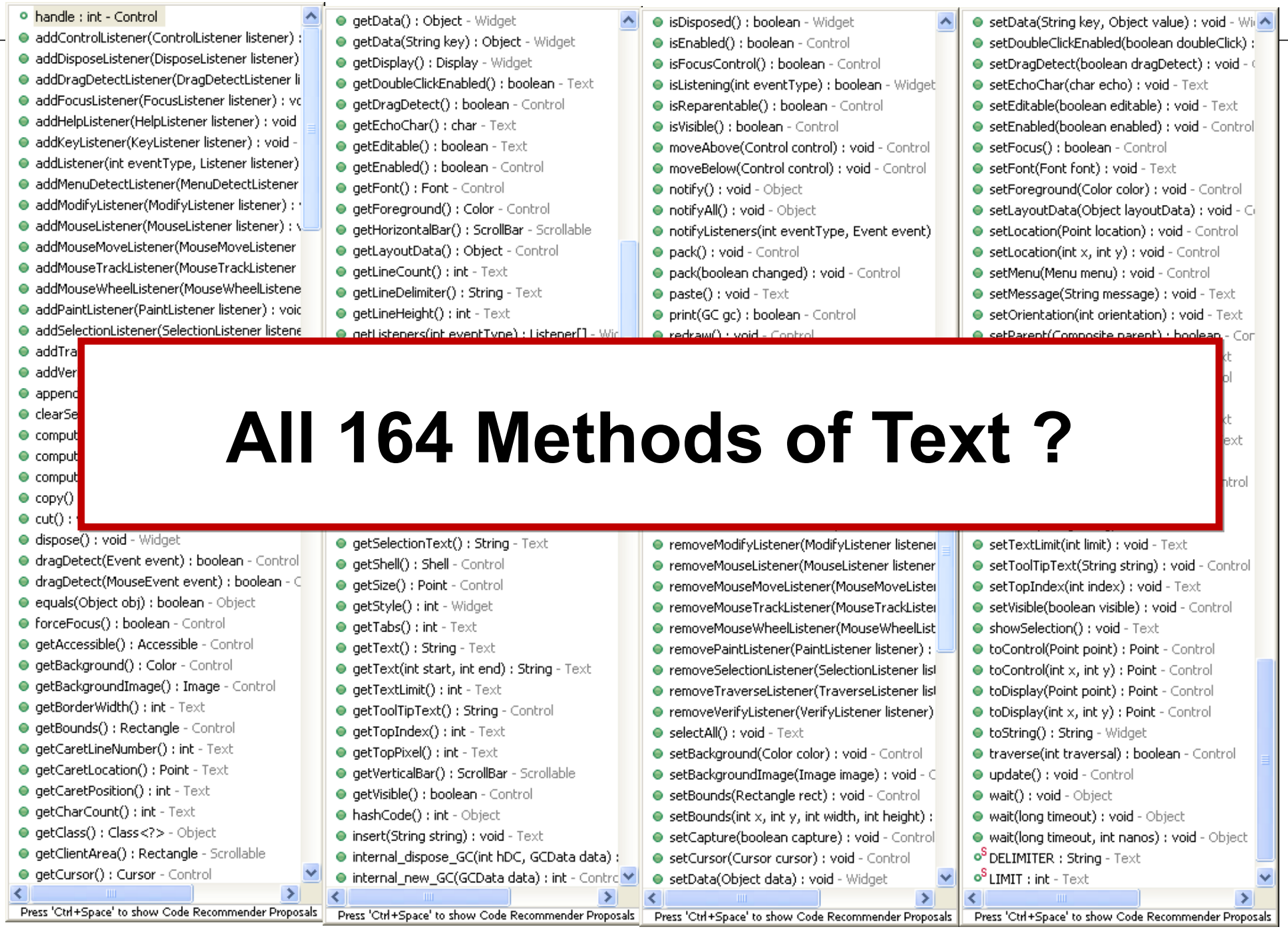
- Automated Method Completion (Hill, Rideout), 2004
- Learning from Examples to Improve Code Completion Systems (Bruch, Monperrus, Mezini), 2009
- Analogy-Based Practical Classification Rules for Software Quality Estimation (Khoshgoftaar and Seliya), 2003
- Estimating Maintenance Effort By Analogy (Leung), 2003

Code completion: what do you need?

```
    @Override
    protected Control createDialogArea(Composite parent) {
        Composite container = (Composite) super.createDialogArea(parent);
        swtTextWidget = new Text(container, SWT.BORDER);
        swtTextWidget.
        return container;
    }
}
```

What should the code completion give?

Motivation



All 164 Methods of Text ?

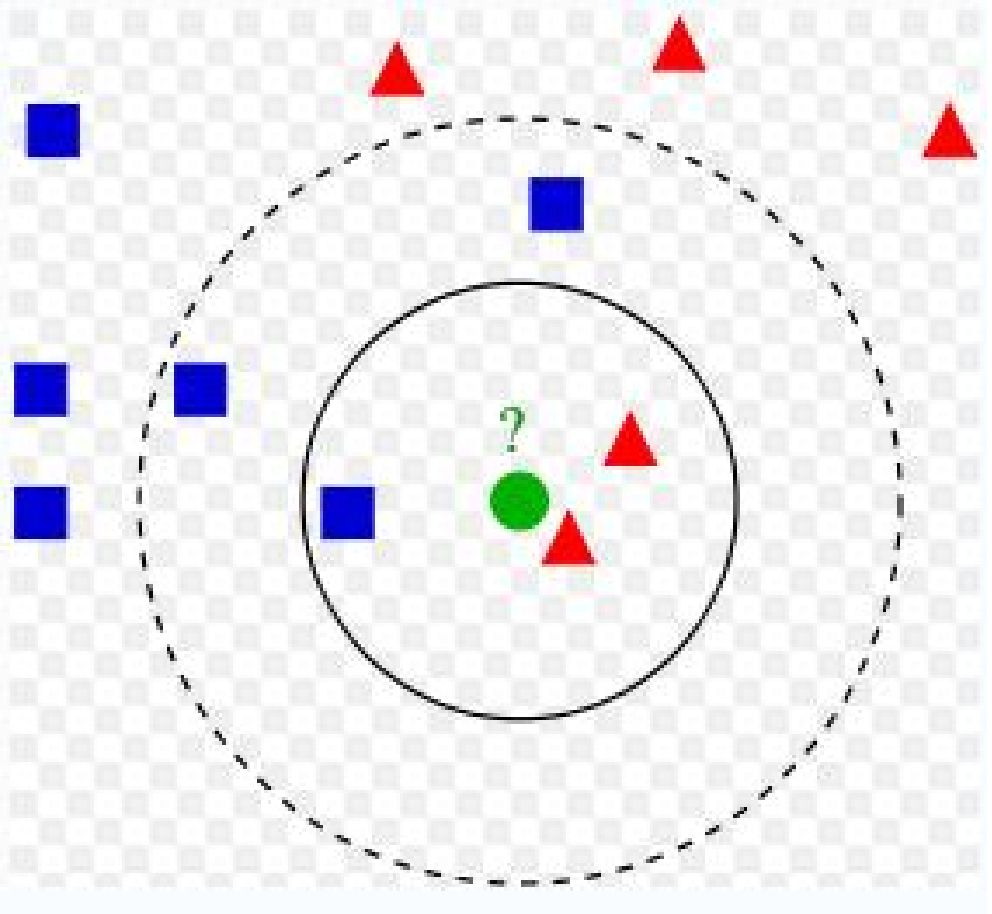
Code completion: what do you need?

```
    @Override
    protected Control createDialogArea(Composite parent) {
        Composite container = (Composite) super.createDialogArea(parent);
        swtTextWidget = new Text(container, SWT.BORDER);
        swtTextWidget.
        return container;
    }
}
```

We can look for similar code and recommend methods that are in similar code.

Similar code = nearest neighbors

Reminder



Task: classifying the observation (in green) as triangle or square

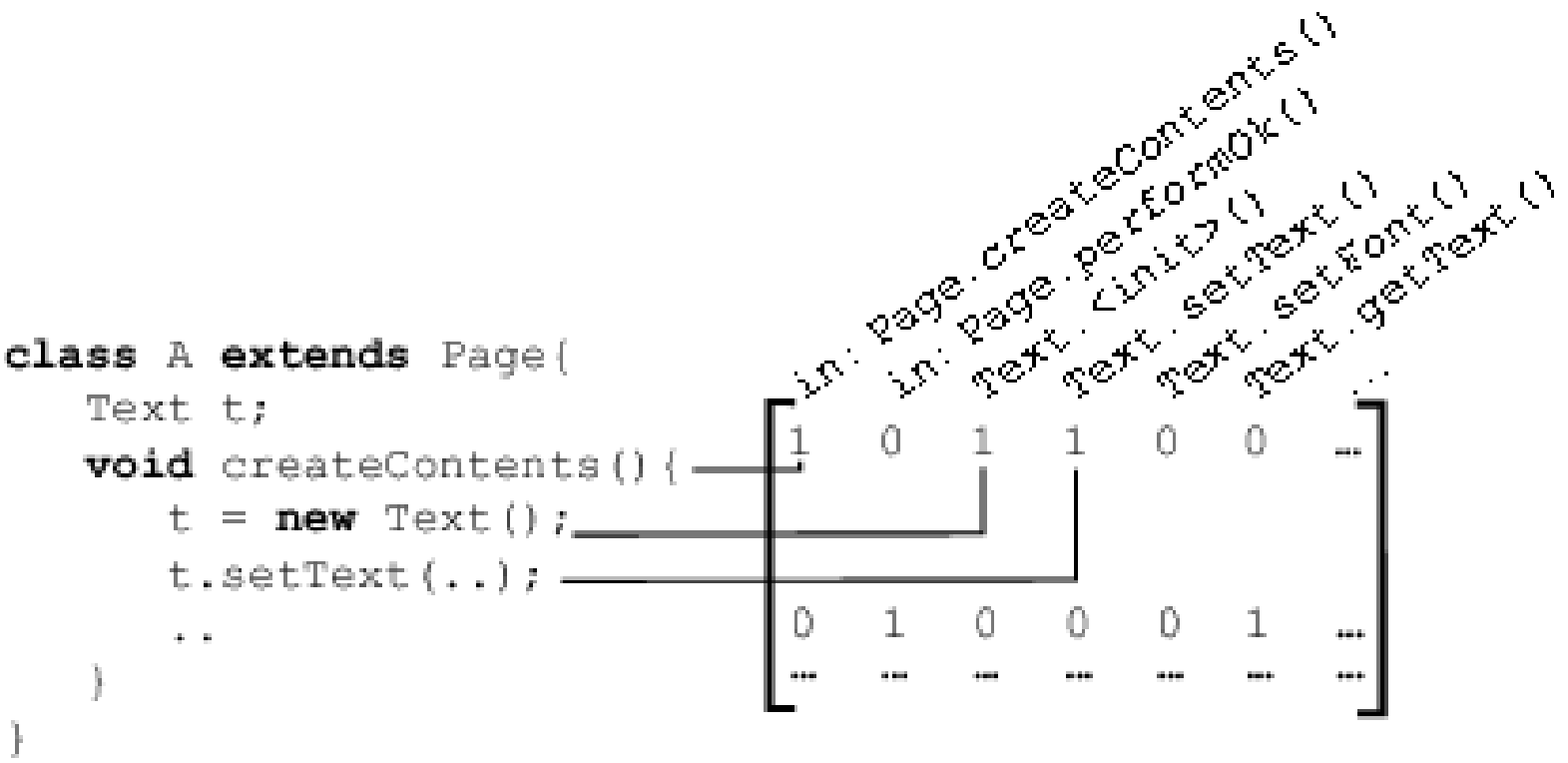
Space: 2-dimensional real features

Metric: euclidean distance

Strategy: majority

See also: Nearest neighbor pattern classification (Cover and Hart), 1967

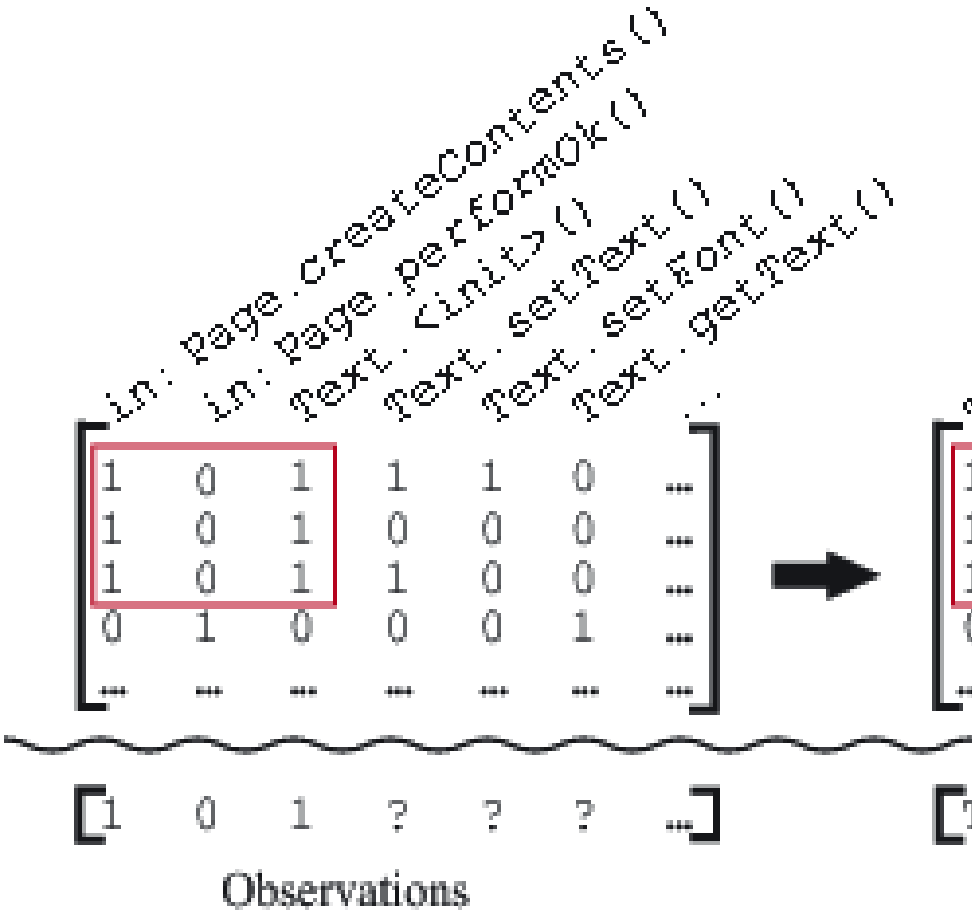
Feature Space



Each variable is a point

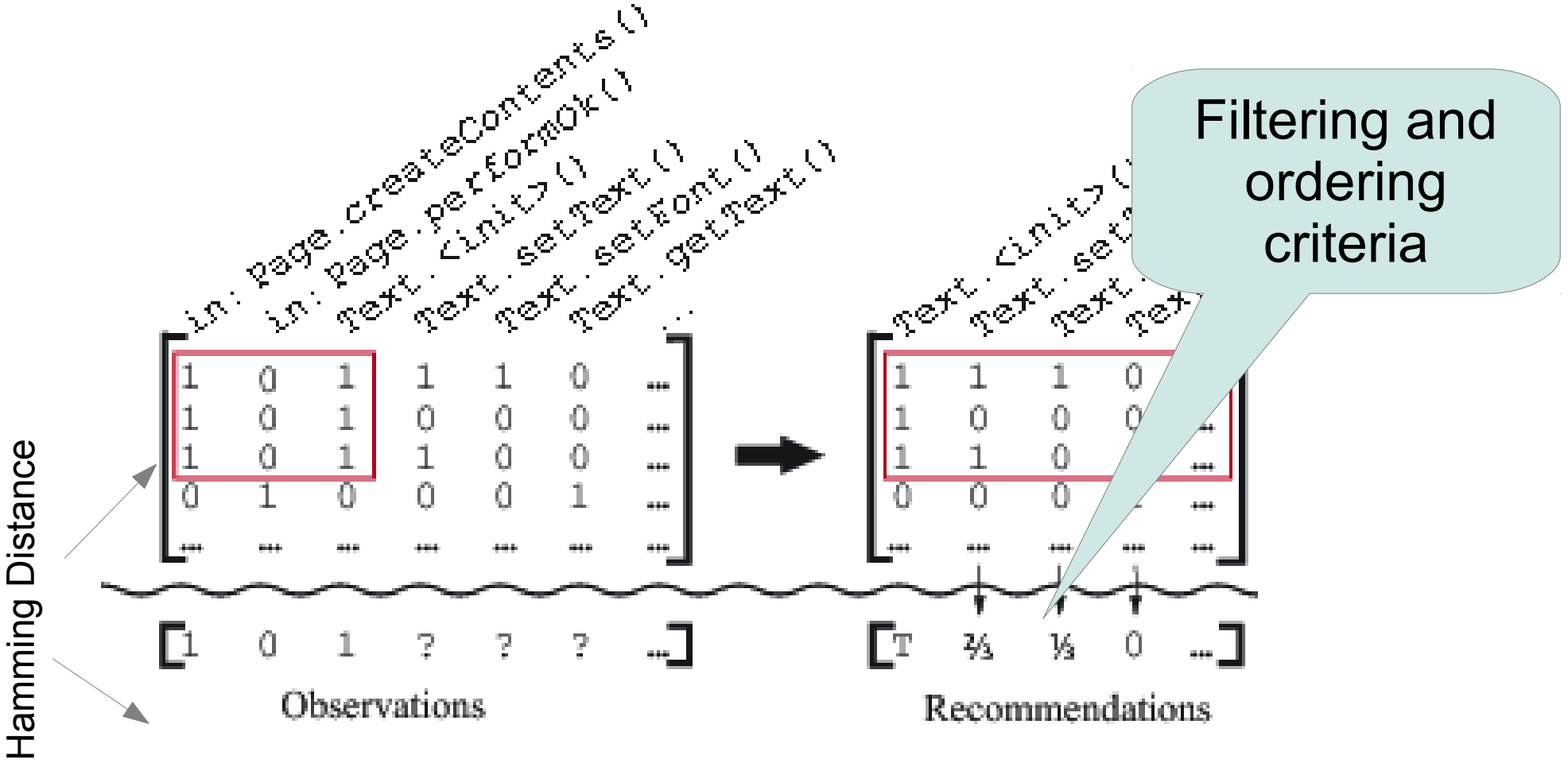
In a binary space of size 1500+

Distance metric



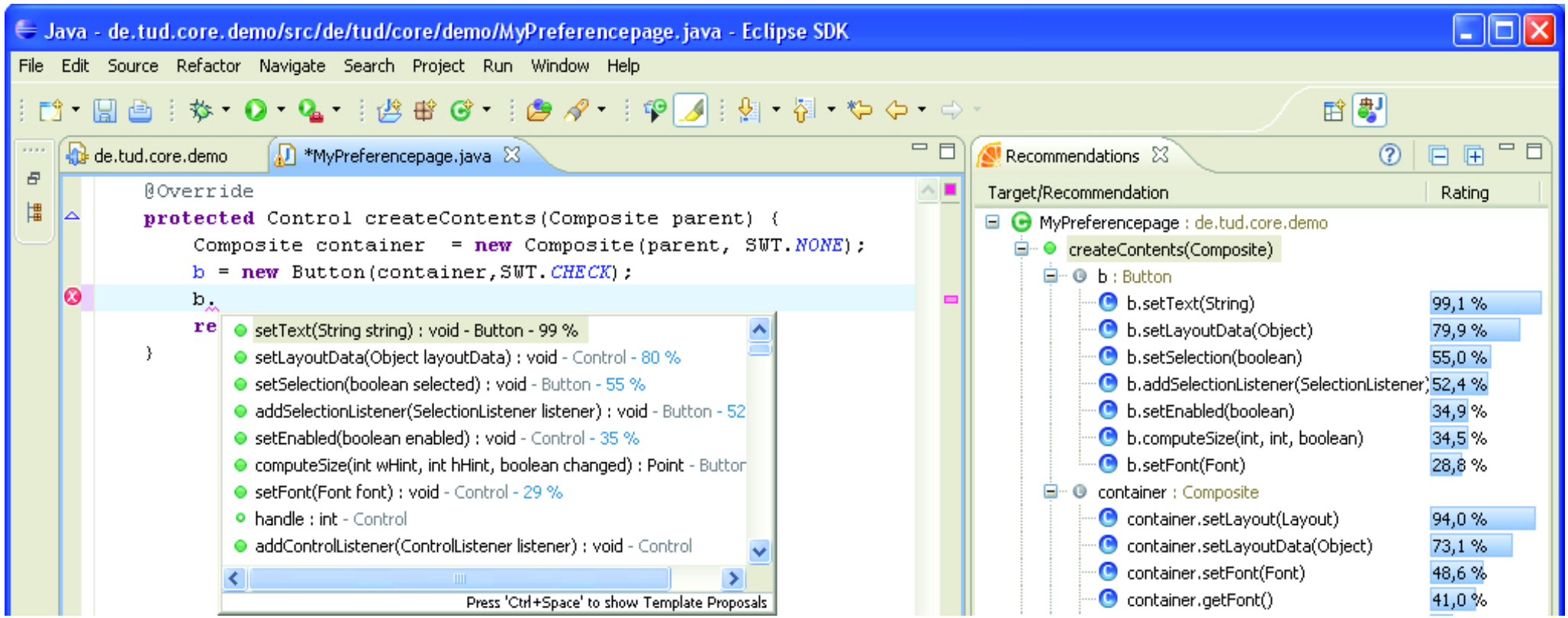
Distance metric: only ones in the observation are taken into accounts

Classification? No, prediction!



No classification but prediction of the calls to be put in the code completion systems based on the nearest neighbors

Screenshot



- ~~Ordered by alphabetical order~~
- ~~Poorly context aware~~
- Too many (useless) methods proposed?

Automatic Evaluation

- simulate queries to code completion based on real data
- arbitrary choice of removing 50% of the calls
- train and test dataset are different (cross-validation)
- large-scale (50000+ different queries)
- precision and recall as performance metrics

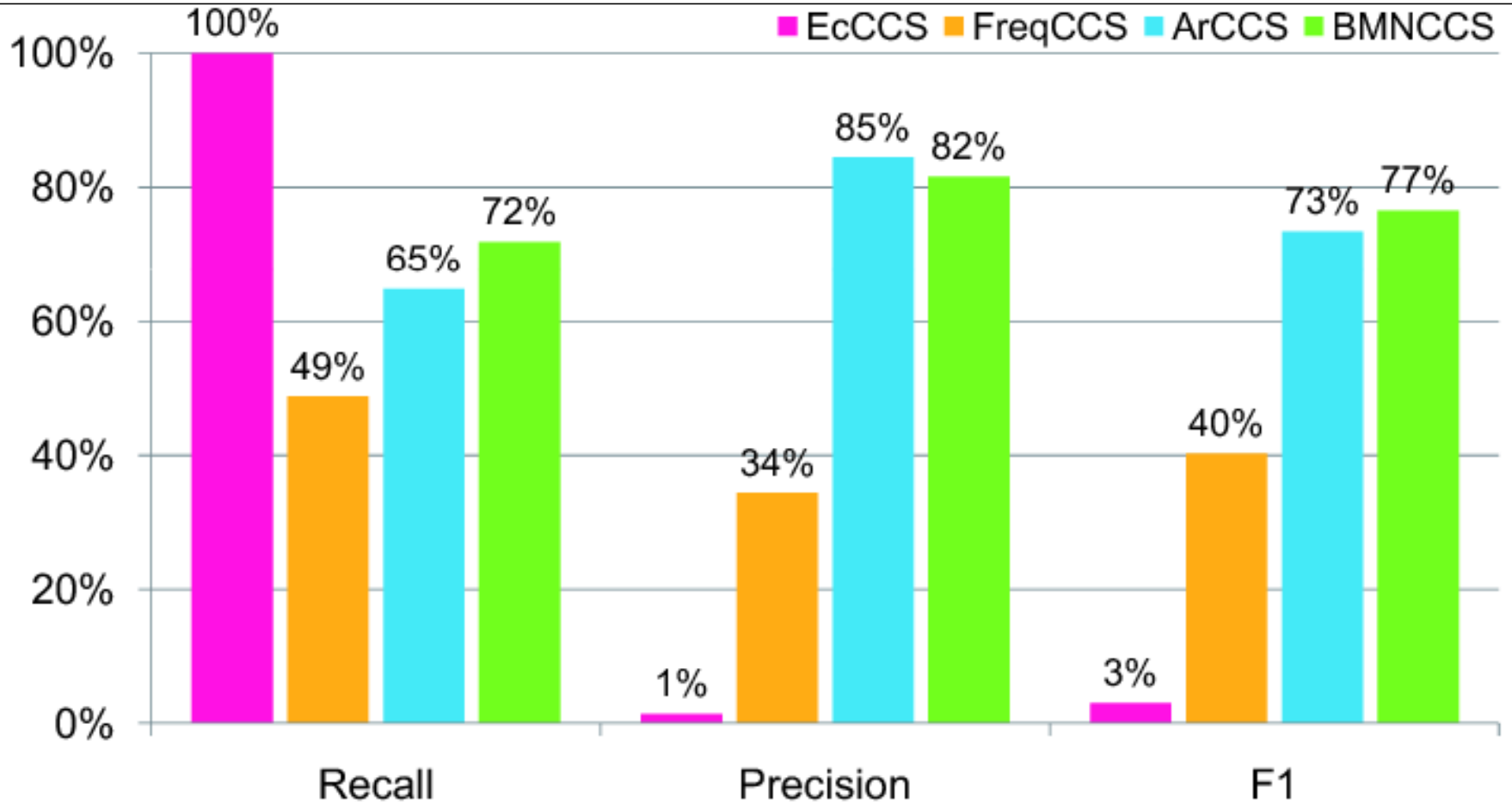
Precision and Recall

- Record: SWT::Text object, <init>, setText, setFont (real data)
- Split in:
  - Query: <init>, setText
  - Expectation: setFont
- Recommended:
  - setFont, getText
- Precision: 50%, Recall: 100%

Competitors

- Type system (default Eclipse code completion)
  - based on the type system, recall = ?
- Frequency based
  - based on the observed frequency with a cut-off threshold
- Association rule based
  - out of the scope of this lecture

Evaluation



Code completion can be improved based on:

- an appropriate feature space (context)
- an appropriate metric

Advertisement

Try it by yourself !

http://www.eclipse.org/recommenders/

Automatic method completion

- Certain code clones are good
- Generally small
- They are small units of implementation, for example implementing a listener interface, or handling an keyboard event

How to find them at development time?

Automatic method completion

- For each method k in the software project, one generates a corresponding v^k of the form:

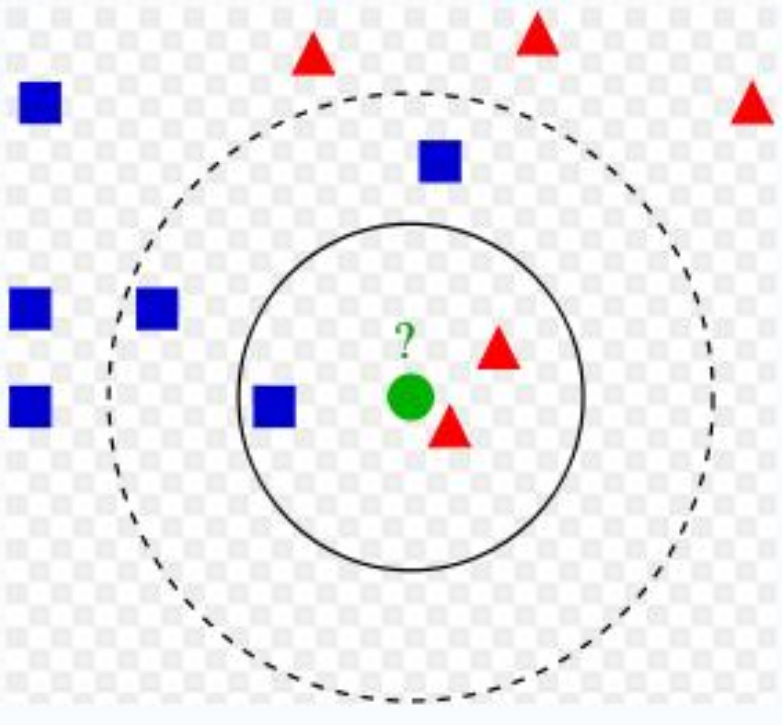
  v^k = [lines, complexity, args, token1 . . . token150]

- Token: count of each of the 150 token types that the Java Language Specification
- Integer space of space 153

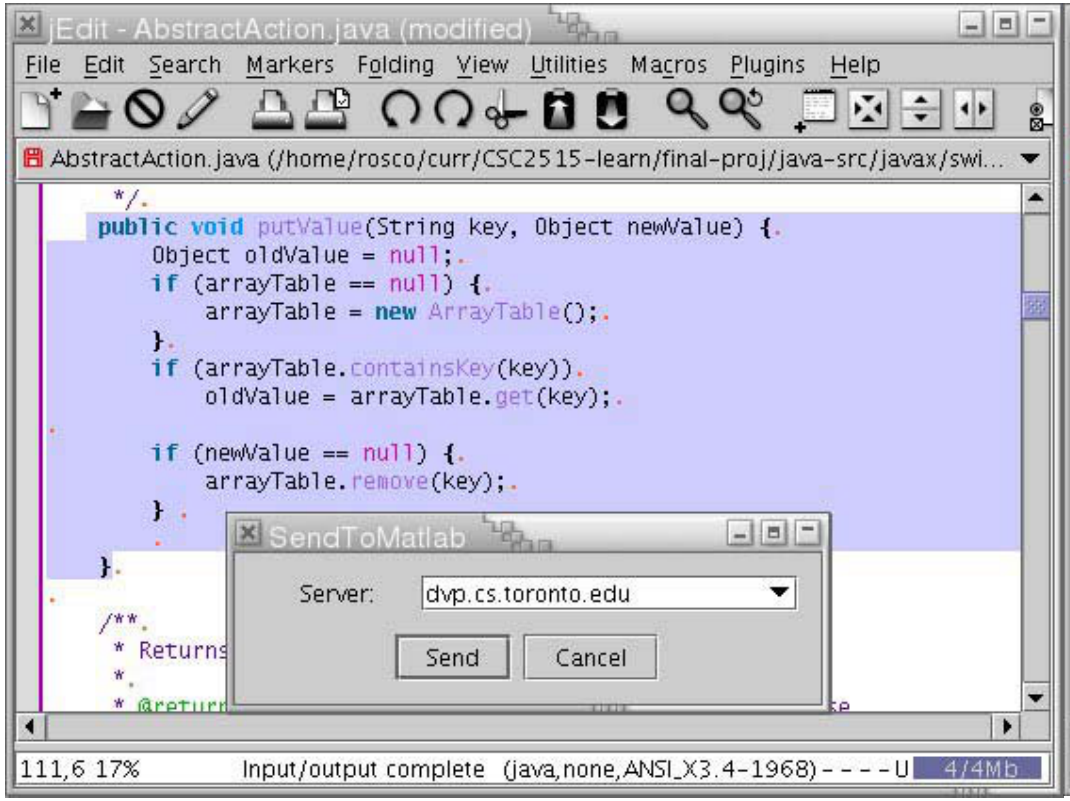
Automatic method completion

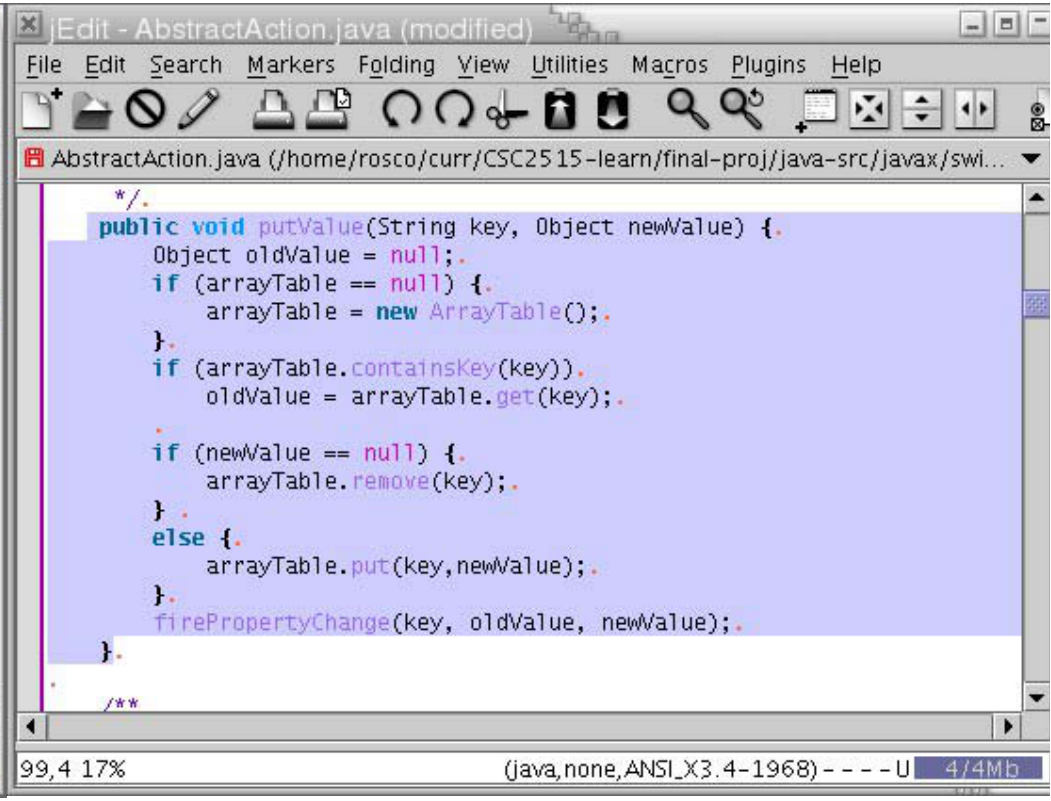
- Euclidean distance
- The number (k) of nearest neighbors does not matter, the RSSE orders by distance

```
protected void fireCaretUpdateTwo(CaretEvent e) {
  // Should I check if listeners is null?
  Object[] listeners = listenerList.getListenerList();
  // Send event to all listeners
  for(int i = listener.length;i>=0;i-=1) {
    ((CaretListener)listeners[i]).caretUpdate(e);
  }
}
```



Jedit prototype



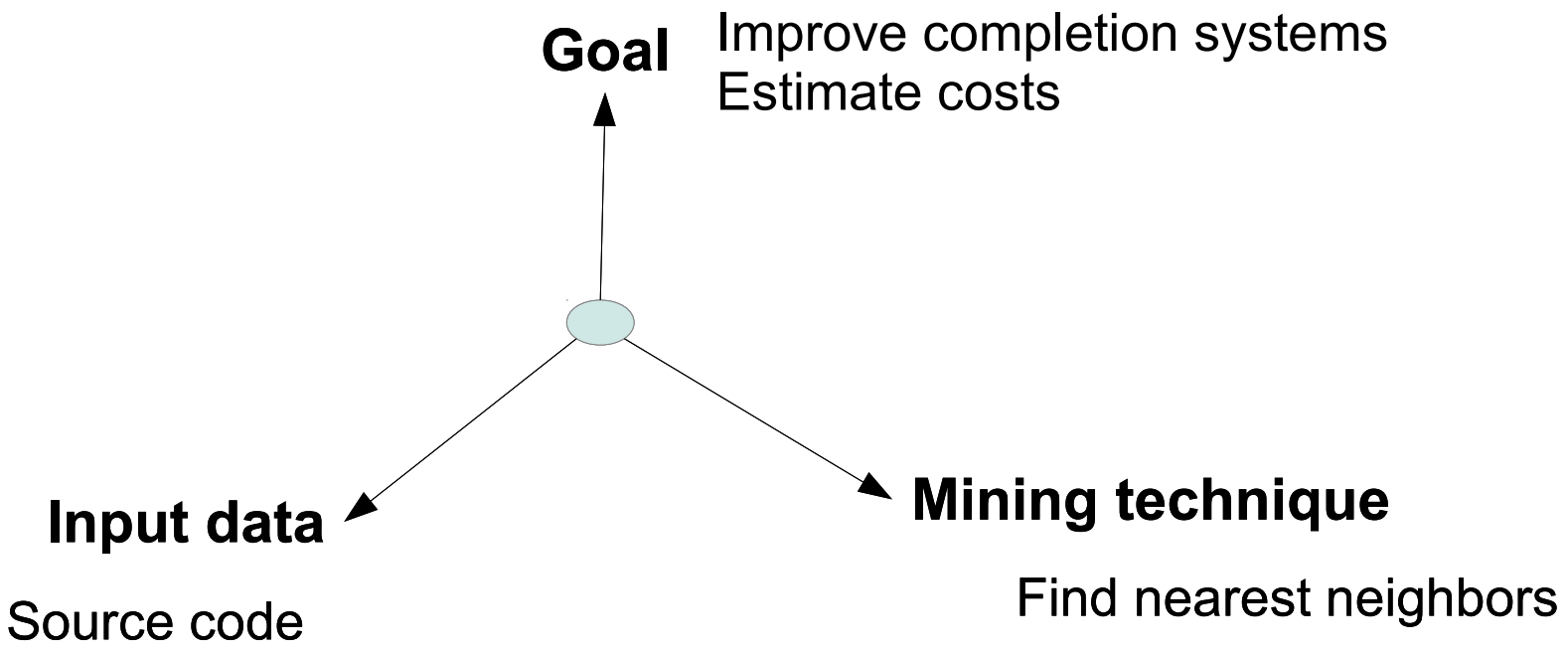


Summary

	Bruch et al, 2009	Hill et al. 2004	other
Goal	method call completion	method body completion	cost estimation
Feature space	binary space, ~1500	integer space, ~150	
Distance metric	hamming based	euclidean	
Classification	none	none	
Synthesis	recommendations	none	

Recap.

- It is powerful to use data-mining techniques for solving software engineering problems
- Technique: nearest neighbors



The big picture

Datamining for Software Engineering

Three axes to characterize the contributions in the field.