Codewebs: Scalable Homework Search for Massive Open Online Programming Courses

Educause, 11/6/2013
Massive Scale Education

Coursera
4.5 million users

Udacity
750,000 users

edX
1.2 million users
Complex and informative feedback in MOOCs

Short Response

- Easy to automate
- Limited ability to ask expressive questions or require creativity

Long Response

- Hard to grade automatically
- Can assign complex assignments and provide complex feedback

Types of assignments:
- Coding assignments
- Proofs
- Essay questions
%GRADIENTDESCENT Performs gradient descent to learn theta
% theta = GRADIENTDESCENT(X, y, theta, alpha, num_iters) updates theta by
% taking num_iters gradient steps with learning rate alpha

m = length(y);
J_history = zeros(num_iters, 1);
for iter = 1:num_iters
    theta = theta - alpha * 1/m * (X' * (X * theta - y));
    J_history(iter) = computeCost(X, y, theta);
end
function [theta, J_history] = gradientDescent(X, y, theta, alpha, num_iters)
    m = length(y);
    J_history = zeros(num_iters, 1);
    for iter = 1:num_iters
        hypo = X*theta;
        newMat = hypo - y;
        trans1 = (X(:,1))';
        trans1 = trans1';
        newMat1 = trans1 * newMat;
        temp1 = sum(newMat1);
        temp1 = (temp1 * alpha)/m;
        A = [temp1];
        theta(1) = theta(1) - A;
        trans2 = (X(:,2))';
        newMat2 = trans2*newMat;
        temp2 = sum(newMat2);
        temp2 = (temp2 * alpha)/m;
        B = [temp2];
        theta(2) = theta(2) - B;
        J_history(iter) = computeCost(X, y, theta);
    end
    theta(1) = theta(1)
    theta(2) = theta(2);
end

Better: theta = theta - (alpha/m) * X' * (X * theta - y)

Correctness: Good
Efficiency: Good
Style: Poor
Elegance: Poor
sum($x' \times (x' \times \theta - y)$);

sum(((\theta' \times x)' - y)' \times x);

sum(transpose(x' \times \theta - y) \times x);

...  

...  

Source code similarity  

Scalability
The Data
Coursera Machine Learning

> 1 million submissions
Why Care?
Fast feedback is key for student success [Dihoff ‘04].

Human feedback is an overwhelming time commitment for teachers [Sadler ‘06].
Cost disease

Direct Costs Per Student, Compared with an Economy-Wide Cost Index

Index, 1904-05 = 100

Fiscal Year, beginning 1904-05

- Chicago-Princeton-Vanderbilt Average Direct Costs Per Student
- Economy-Wide Cost Index
Grand Challenge

Automatically provide feedback to students (for programming assignments)
Basic Idea
Input: many ungraded submissions for a programming assignment
[Effective Teacher]

Force multiply teacher effort.
Build A Homework Search Engine
function A = warmUpExercise()
    A = [];
    A = eye(5);
endfunction

ASTs ignore:
- Whitespace
- Comments
- ...
Abstract syntax tree representations

```plaintext
function A = warmUpExercise()
    A = [];
    A = eye(5);
endfunction
```

ASTs ignore:
- Whitespace
- Comments
- ...
Indexing documents by phrases

What basic queries should code search engine support?

<table>
<thead>
<tr>
<th>term/phrase</th>
<th>document list</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td>{2,5,6,9,56}</td>
</tr>
<tr>
<td>sky</td>
<td>{1,2}</td>
</tr>
<tr>
<td>submarine</td>
<td>{2,3,4}</td>
</tr>
<tr>
<td>woes</td>
<td>{10,19,38}</td>
</tr>
<tr>
<td>yellow</td>
<td>{2,4}</td>
</tr>
</tbody>
</table>

The bright and blue butterfly hangs on the breeze…

We all something something yellow submarine…

"blue sky"

"yellow submarine"
Find the Feet That Fit

Query

ASTs that match on context
Find the Feet That Fit

ASTs that match on context
Running time for indexing

- Top graph: Runtime (seconds) vs. # ASTs indexed.
- Bottom graph: Runtime (seconds) vs. Average AST size (# nodes).

- Runtime increases with the number of ASTs indexed.
- Average AST size also increases with longer runtime.

The data shows a clear trend indicating that the time required for indexing grows with the increase in the number of ASTs and their complexity.
Zipf’s Law for code phrases

Subtree frequency

Subtree rank

$\log(\text{rank}) \approx 10.7 - 1.01 \log(\text{freq})$

“starter code elbow”
Great!
So what?
Data driven equivalence classes of code phrases
There is Structure to Code
def doSomething() {

Part A  500 ways
Part B  200 ways
Part C  200 ways

Without factoring:
\[ \text{params} = 20 \times 10^6 \]

With factoring:
\[ \text{params} = 9 \times 10^2 \]
Say you have two sub-forests...
Each sub-forest has a complement

Solution A

Solution B
If those complements are equal...
And the programs have the same output…

Solution A

Solution B
The sub-forests were analogous in that context.
\[ \theta = (X'X)^{-1}X'y \]

\[ \text{sum}(X_*\text{repmat}(\theta',\{m\},1),2) \]

\[ \alpha = \frac{1}{m} \]

\[ \text{sum}(\text{hypothesis} - y, 2) \]
Reduction in #ASTs via canonicalization

More equivalence classes means more reduction means fewer ASTs

We get better results on the more common ASTs

(Ordered by frequency of AST)
How many submissions can we give feedback to with fixed effort?

Number of submissions covered (out of 40,000)

- With 25 ASTs marked
- With 200 ASTs marked

Number of equivalence classes

0, 1, 10, 19
How many submissions can we give feedback to with fixed effort?

- With 25 ASTs marked:
  - Equivalences +
  - No Equivalences +

- With 200 ASTs marked:
  - Equivalences +
  - No Equivalences +

Number of submissions covered (out of 40,000)

- # equivalence classes:
  - 0
  - 1
  - 10
  - 19
Syntactic bug isolation using Codewebs
Bug detection evaluation
(Difficult to evaluate localization!!)

Each point represents a single coding problem in Coursera’s ML class

Regularized backpropagation for neural nets
Bug detection evaluation

- **F-score**
- Higher is better

![Graph showing F-score vs. # unique ASTs considered](image)

- **With canonicalization**
- **Without canonicalization**

The graph illustrates the F-score for bug detection evaluation, comparing the performance with and without canonicalization. As the number of unique ASTs considered increases, the F-score generally decreases, indicating a trade-off between the number of ASTs and the accuracy of bug detection. The line with canonicalization stays consistently higher than the line without canonicalization, suggesting that canonicalization improves the F-score.
Concrete Example
Case study: The “extraneous sum”

Syntax based approach:

Attach this message to everyone containing that exact expression (covers 99 submissions)

Correct

Incorrect

Dear Lisa Simpson, consider the dimension of the expression: \( x'\times(x\times\theta-y) \) and what happens after you call sum on it...
The extraneous sum bug takes many forms...

\[
\theta = \theta - \alpha \frac{1}{m} \sum (x' \cdot (x \cdot \theta - y));
\]

\[
\theta = \theta - \alpha \frac{1}{m} \sum ((\theta' \cdot x')' - y)' \cdot x);
\]

\[
\theta = \theta - \alpha \frac{1}{m} \sum (\text{transpose}(x \cdot \theta - y) \cdot x);
\]

(Easier) Output based approach:

Attach this message to everyone containing that exact expression (covers 1091 submissions)
theta = theta - alpha \times \frac{1}{m} \times \sum (X' \times (X \times \theta - y));

Step 1: Find equivalent ways of writing buggy expression using Codewebs engine
Step 2: Write a thoughtful/meaningful hint or explanation
Step 3: Propagate feedback message to any submission containing equivalent expression

~47% improvement over just using an output based feedback system!!

Output based: 1091
Codewebs: 1208
Combined: 1604

# submissions covered by single message
• **Big data as a problem**: can we give human quality feedback to a *million* code submissions?

• **Big data as a solution**: structure (clustering, subtree equivalence) of the solution space invisible without dense sampling!

• **Where we’re going next**:
  • Extensions to other languages
  • Applications to new problem domains
  • Data association problem for variables
  • Dynamic analysis
  • Temporal analysis
How do solutions evolve – *modeling the human creative process*

Feedback on progress, not just on final solution

Think beyond computer science, beyond education