CrystalBLEU: Precisely and Efficiently Measuring the Similarity of Code

Aryaz Eghbali, Michael Pradel
Software Lab, University of Stuttgart
Motivation

Me 😊

Michael 😊

ctrl+c  ctrl+v

---


**BLEU: a Method for Automatic Evaluation of Machine Translation**

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA
{papineni,roukos,toddward,weijing}@us.ibm.com
BLEU

Designed for Natural Language (spec. machine translation)

\[ p_n = \frac{\sum_{C \in \text{Candidates}} \sum_{ngram \in C} \text{Count}_{\text{clip}}(ngram)}{\sum_{C' \in \text{Candidates}} \sum_{ngram' \in C'} \text{Count}(ngram')} \]

maximum length
of n-grams

\[ \text{BLEU} = \exp\left(\min\left(1 - \frac{r}{c}, 0\right) + \sum_{i=1}^{\max N} w_i \log p_i\right) \]

brevity penalty
weighted average of modified precisions

BLEU

- Fast
- Works for partial code
- Language agnostic
- Similar-looking different code

// Reference:
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int t = in.nextInt();
        in.nextLine();
        while (t-- > 0 ) {
            System.out.println(new StringBuffer(in.nextLine()).reverse());
        }
    }
}

// Hypothesis 1: equivalent to the reference
import java.util.Scanner;
public class Main {
    public static void main(String argv[]) {
        int num_of_tests = 0;
        Scanner in = new Scanner(System.in);
        num_of_tests = Integer.parseInt(in.nextLine());
        for(int i=0; i<num_of_tests; i++) {
            StringBuilder rev_str = new StringBuilder(in.nextLine());
            System.out.println(rev_str.reverse());
        }
    }
}

// Hypothesis 2: not equivalent to the reference
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNext()) {
            System.out.println(in.nextInt() + in.nextInt());
        }
    }
}
Difference between PL & NL

Syntax, coding conventions, etc.

→ Common n-grams

→ Trivially shared n-grams
Most Common N-grams

Most frequent n-grams in PL are more frequent that the most frequent n-grams in NL

<table>
<thead>
<tr>
<th></th>
<th>2-grams</th>
<th>% of 2-grams</th>
<th>4-grams</th>
<th>% of 4-grams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>) ;</td>
<td>5.49</td>
<td>) ; ; )</td>
<td>1.34</td>
</tr>
<tr>
<td></td>
<td>( )</td>
<td>4.75</td>
<td>( ) {</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>) {</td>
<td>3.83</td>
<td>( ) ;</td>
<td>1.14</td>
</tr>
<tr>
<td></td>
<td>; ;</td>
<td>3.81</td>
<td>) ) ;</td>
<td>1.12</td>
</tr>
<tr>
<td></td>
<td>; import</td>
<td>2.75</td>
<td>) ; } public</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>) )</td>
<td>1.73</td>
<td>( ) ) ;</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>} public</td>
<td>1.27</td>
<td>) { this</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>{ return</td>
<td>1.24</td>
<td>; } public void</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>} }</td>
<td>1.07</td>
<td>; } @Override public</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>) .</td>
<td>0.99</td>
<td>) { if</td>
<td>0.54</td>
</tr>
<tr>
<td>English</td>
<td>of the</td>
<td>2.31</td>
<td>”, he said</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>, and</td>
<td>1.45</td>
<td>, he said ,</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td>in the</td>
<td>1.31</td>
<td>, of course ,</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>” .</td>
<td>1.01</td>
<td>” , I said</td>
<td>0.29</td>
</tr>
<tr>
<td></td>
<td>, the</td>
<td>0.85</td>
<td>” , she said</td>
<td>0.29</td>
</tr>
<tr>
<td></td>
<td>to the</td>
<td>0.85</td>
<td>, he said .</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>“ .</td>
<td>0.64</td>
<td>” . “ I</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>” ,</td>
<td>0.63</td>
<td>he said , “</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>on the</td>
<td>0.57</td>
<td>” ? asked .</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>, but</td>
<td>0.53</td>
<td>, I said .</td>
<td>0.19</td>
</tr>
</tbody>
</table>
Unrelated Pairs of Text

Two PL texts share more n-grams than two NL texts
CrystalBLEU

- Extract common n-grams (as trivially shared n-grams)
- Run BLEU, ignoring the top K (~500) common n-grams
- Similar to stop words in NLP

Function `modified_precision(ref, hyp, i, S)` is

```plaintext
refCounts ← n-grams of length i from each ref and their number of occurrences
hypCount ← n-grams of length i from hyp and their number of occurrences
remove any n-grams from refCounts and hypCount that is in S, or divide refCounts and hypCount by the logarithm of counts in S
for ngram ∈ hypCount do
    clipped_countngram ← min(hypCountngram, max(refCountsngram))
end
numerator ← \( \sum_i clipped\_count_i \)
denominator ← max(1, \( \sum \) hypCount)
return numerator, denominator
```

similarity score
Distinguishability

Measure how well a metric can distinguish similar and dissimilar pairs

Ratio of similarity within classes to between classes

\[ d = \frac{m(\text{Pairs}_{\text{intra}})}{m(\text{Pairs}_{\text{inter}})} \]
Research Questions

RQ1: How well does CrystalBLEU distinguish similar and dissimilar programs?

RQ2: Can CrystalBLEU avoid misleading comparisons?

RQ3: How efficient is calculating CrystalBLEU?
RQ1: Distinguishability

ShareCode online judge

Human written code

Accepted solutions to a problem → Class of equivalent programs

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
<th>CodeBLEU</th>
<th>CrystalBLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra-class</td>
<td>0.79</td>
<td>0.52</td>
<td>0.65</td>
</tr>
<tr>
<td>Inter-class</td>
<td>0.32</td>
<td>0.36</td>
<td>0.10</td>
</tr>
<tr>
<td>Distinguishability</td>
<td>2.47</td>
<td>1.44</td>
<td>6.50</td>
</tr>
</tbody>
</table>
RQ1: Clone Detection

BigCloneBench

Simple threshold-based classification

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
<th>CrystalBLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.66</td>
<td>0.82</td>
</tr>
<tr>
<td>Precision</td>
<td>0.20</td>
<td>0.37</td>
</tr>
<tr>
<td>Recall</td>
<td>0.52</td>
<td>0.46</td>
</tr>
<tr>
<td>F1 score</td>
<td>0.20</td>
<td>0.37</td>
</tr>
</tbody>
</table>
**RQ2: Evaluating Code Generators**

**Ground truth**

```java
Rational function ( int arg0 ) { VideoTrack loc0 = tracks . get ( arg0 ) ; return new Rational ( loc0 . count , 30 ) ; }
void function () { return this . released ; }
void function ( ) { SecurityConfiguration . getApplicationPolicy ( "string" ) ; }
boolean function ( ) { return ( type == DICTIONARY ) ; }
void function ( List = Integer = arg0 ) { takeLock . lock ( ) ; try { taskIdsQueue . add ( arg0 ) ; notEmpty . signal ( ) ; } finally { takeLock . unlock ( ) ; } }
void function ( boolean arg0 ) { fNsLevelBindings = arg0 ; }
boolean function ( ) { return us . fNsAll ( locator ) . length == 1 ; }
boolean function ( ) { return ( type == DICTIONARY ) ; }
void function ( int arg0 ) { list . level = arg0 ; }
void function ( int arg0 ) { list [ ] . loc0 = extractKeys ( arg0 ) ; for ( int loc1 = loc0 . length + 1 ; loc1 <= 0 ; -- loc1 ) doKeyUp ( loc0 [ loc1 ] ; ) ; }

int function ( ) { return 0 ; }
int function ( ) { return release ; }
void function ( ) { SecurityConfiguration . getApplicationPolicy ( ) ; }
boolean function ( ) { return type == NAME ; }
void function ( List = Integer = arg0 ) { taskIdsQueue . addAll ( arg0 ) ; }
void function ( boolean arg0 ) { fNsBindingsRecovery = arg0 ; }
boolean function ( ) { return locator == null ; }
boolean function ( ) { return type == NAME ; }
void function ( int arg0 ) { task . level = arg0 ; }
void function ( KeyEvent arg0 ) { }
```  

**Neural model**

**Dummy model**

Neural Model: Concode*

Dummy Model: Generate code mostly consisting of common n-grams

---

CrystalBLEU

// Reference:
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int t = in.nextInt();
        in.nextLine();
        while (t-- > 0) {
            System.out.println(new StringBuffer(in.nextLine()).reverse());
        }
    }
}

// Hypothesis 1: equivalent to the reference
import java.util.Scanner;
public class Main {
    public static void main(String argv[]) {
        int num_of_tests = 0;
        Scanner in = new Scanner(System.in);
        num_of_tests = Integer.parseInt(in.nextLine());
        for (int i = 0; i <= num_of_tests; i++) {
            StringBuilder rev_str = new StringBuilder(in.nextLine());
            System.out.println(rev_str.reverse());
        }
    }
}

// Hypothesis 2: not equivalent to the reference
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNext()) {
            System.out.println(in.nextInt() + in.nextInt());
        }
    }
}
## RQ3: Runtime

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of tokens</th>
<th>Preprocessing time (s)</th>
<th>BLEU</th>
<th>CodeBLEU</th>
<th>CrystalBLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShareCode Java</td>
<td>580K</td>
<td>4.8</td>
<td>1036.9</td>
<td>5382.3</td>
<td>953.6</td>
</tr>
<tr>
<td>ShareCode C++</td>
<td>1.8M</td>
<td>19.9</td>
<td>868.9</td>
<td>3848.3</td>
<td>743.6</td>
</tr>
<tr>
<td>BigCloneBench</td>
<td>2.6M</td>
<td>22.3</td>
<td>83.5</td>
<td>1445.1</td>
<td>85.7</td>
</tr>
<tr>
<td>Concode (tokenized)</td>
<td>2.6M</td>
<td>4.1</td>
<td>81.5</td>
<td>1269.4</td>
<td>81.7</td>
</tr>
</tbody>
</table>
How to Use

Install:

> pip install crystalbleu

Use:

```python
from collections import Counter
# Import CrystalBLEU
from crystalbleu import corpus_bleu

# Extract trivially shared n-grams
k = 500
frequencies = Counter(tokenized_corpus)  # tokenized_corpus is a
# list of strings
trivially_shared_ngrams = dict(frequencies.most_common(k))

# Calculate CrystalBLEU
crystalBLEU_score = corpus_bleu(
    references, candidates, ignoring=trivially_shared_ngrams)
```
CrystalBLEU Features

- As fast as BLEU
  - Faster than CodeBLEU
- Works on partial code
- Language agnostic
- Higher distinguishability

<table>
<thead>
<tr>
<th>Property</th>
<th>BLEU</th>
<th>CodeBLEU</th>
<th>RUBY</th>
<th>CrystalBLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language-agnostic</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
</tr>
<tr>
<td>Handle incomplete and partially incorrect code</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
</tr>
<tr>
<td>Efficient</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
</tr>
<tr>
<td>High distinguishability</td>
<td>❌</td>
<td>❌</td>
<td>N/A</td>
<td>✓</td>
</tr>
</tbody>
</table>
Q&A

CrystalBLEU

Install:

> pip install crystalbleu

Use:

```python
from collections import Counter
# Import CrystalBLEU
from crystalbleu import corpus_bleu

# Extract trivially shared n-grams
k = 500
frequencies = Counter(tokenized_corpus)  # tokenized_corpus is a list of strings
trivially_shared_ngrams = dict(frequencies.most_common(k))

# Calculate CrystalBLEU
crystalBLEU_score = corpus_bleu(  
    references, candidates, ignoring=trivially_shared_ngrams)
```

<table>
<thead>
<tr>
<th>Property</th>
<th>BLEU</th>
<th>CodeBLEU</th>
<th>RUBY</th>
<th>CrystalBLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language-agnostic</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
</tr>
<tr>
<td>Handle incomplete and partially incorrect code</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
</tr>
<tr>
<td>Efficient</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
</tr>
<tr>
<td>High distinguishability</td>
<td>✗</td>
<td>✗</td>
<td>N/A</td>
<td>✅</td>
</tr>
</tbody>
</table>

Aryaz Eghbali (aryaz.egh@gmail.com)

Software Lab (https://software-lab.org/)

Code @ https://github.com/sola-st/crystalbleu
RQ4: Parameter K
Datasets

- English: Brown
- French: Europarl
- Java: Java-small
- Python: py150k
- C++: POJ-104
- Equivalent by behavior: ShareCode
- Equivalent by label: BigCloneBench
- Neural model predictions: Concode
## Features

Small human study (1 subject)

<table>
<thead>
<tr>
<th>Property</th>
<th>BLEU</th>
<th>CodeBLEU</th>
<th>RUBY</th>
<th>CrystalBLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language-agnostic</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Handle incomplete and partially incorrect code</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Efficient</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Correlate well with human judgment</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>High distinguishability</td>
<td>✗</td>
<td>✗</td>
<td>N/A</td>
<td>✔</td>
</tr>
</tbody>
</table>