Quality metrics in AI-Models' output: Code-BLEU Score

Saeed Siddik
Assistant Professor
IIT University of Dhaka

CodeBLEU

- a code-specific evaluation metric that extends BLEU
- lexical n-gram matching (BLEU) with
  - weighted n-gram matching (identifier-aware),
  - syntactic matching (AST-based),
  - data-flow matching (semantic).

Components & formula

$$\text{CodeBLEU} = w_{NG} \cdot NG + w_{WNG} \cdot WNG + w_{AST} \cdot AST + w_{DF} \cdot DF$$

- `NG` = n-gram BLEU score (standard modified BLEU)
- `WNG` = weighted n-gram score (gives higher weight to important tokens like identifiers)
- `AST` = syntax / AST match score (how similar parsed ASTs are)
- `DF` = data-flow match score (how similar variable/data flow is)

Example

Reference code (human):

```
def add(a, b):\n return a + b
```

Candidate code (AI):

```
def add(x, y):\n return x + y
```

Compute NG (BLEU)

- **Tokenize** (simple tokenization):
  - Reference tokens: `[def, add, (, a, ,, b, ), :, return, a, +, b]` → 12 tokens
  - Candidate tokens: `[def, add, (, x, ,, y, ), :, return, x, +, y]` → 12 tokens
- **Unigram overlap**: tokens matching by literal string: `def, add, (, ,, ), :, return, +` = **8 matches** (note: `a ≠ x, b ≠ y`).

- Unigram precision = matches / candidate_unigrams = `8 / 12 =` **`0.66`**

Reference code (human):

```
def add(a, b):\n return a + b
```

Candidate code (AI):

```
def add(x, y):\n return x + y
```

Bigram overlap

- Reference bigrams (11):

```
def add, add (, ( a, a ,, , b, b ), ) :, :
  return, return a, a +, + b
```

- Candidate bigrams (11):

```
def add, add (, ( x, x ,, , y, y ), ) :, :
  return, return x, x +, + y
```

- Overlapping bigrams:

```
def add, add (, ) :, : return
```

4 matches out of 11

- Bigram precision = `4 / 11 ≈ 0.36`

BLUE Score Calculation

- Geometric mean = $\sqrt{(0.67 \times 0.36)} = 0.49$
- Since, candidate & reference lengths equal → Brevity Penalty $BP = 1$.

- So NG (BLEU-like n=1..2) ≈ **0.4924.**

Compute WNG (weighted n-gram)

- Weighted n-gram increases weight for identifiers (a,b in reference; x,y in candidate).
- Since identifiers differ, weighted unigram overlap for identifiers = 0. But keywords/punctuation match.
- Let's assume a simple weighted scheme where identifiers get weight 2.0 and other tokens weight 1.0. For this pair:

Compute WNG (weighted n-gram)

- Total weighted matches (candidate perspective): matched tokens and their weights:
- `def (1), add (1), ( (1), , (1), ) (1), : (1), return (1), + (1)`
- total weight = 8 × 1 = 8
- Candidate total weighted tokens = sum of weights of candidate tokens = identifiers `x, y` would be `2` each, others 1 each
- candidate weights = 10 others*1* + *2 identifiers*2 = 10*1* + *2*2 = 10 + 4 = 14 (counting exact tokens: 12 tokens with identifiers weighted higher)

Compute WNG (weighted n-gram)

- Weighted precision = matched_weight / candidate_total_weight `= 8 / 14 = 0.57`

- **WNG ≈ 0.5714.**

Compute AST match score

- AST matching compares the parsed tree shapes and node types
- For `def add(a,b): return a + b`

vs

`def add(x,y): return x + y`

- the AST structure and node types are identical (function name, params, return binary op), only identifier names differ.

- A typical AST match score measures structural overlap and gives a high score when structure is identical. We'll set **AST = 1.00** (perfect structural match)

Compute Data-Flow (DF) match score

- Data-flow looks at how values flow (which variables used, dependency graph).
- Here variable names differ but the flow (param → return expression) is identical.
- We'll pick **DF = 1** to reflect near-perfect dataflow equivalence.

Combine into CodeBLEU

- NG = 0.49
- WNG = 0.57
- AST = 1.0
- DF = 1.0

CodeBLEU (equal weights 0.25 each)

- Compute step-by-step using w = 0.25 each:
- `NG = 0.49`
- `WNG = 0.57`
- `AST = 1.0`
- `DF = 1.0`
- **`CodeBLEU ≈ 0.765`**
- CodeBLEU judges this a good result for code generation.

Practice

- reference `def inc(a): return a + 1`
- candidate `def inc(x): return x + 1`

End of CodeBLUE Score