

CodeBLEU calculation:



Example: Simple Addition Function

Reference Code (Correct):

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

Generated Code (AI Output):

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

🔎 How Each Score is Calculated:

1 **BLEU Score** (Text Similarity)

What it does: Compares words/tokens between codes

Step-by-step:

Reference tokens: ['def', 'add', '(', 'a', ',', 'b', ')', ':', 'return', 'a', '+', 'b']
Generated tokens: ['def', 'add', '(', 'x', ',', 'y', ')', ':', 'return', 'x', '+', 'y']

Matching tokens: def, add, (, ,), :, return, +

Total matches: 8 out of 12

1-gram precision = 8/12 = 0.67

2-gram matches: ['def add', 'add (', '(x', ...] vs reference

3-gram matches: ['def add (', 'add (x', ...]

4-gram matches: ...

BLEU = geometric mean of all n-gram precisions

BLEU ≈ 0.65

...

****Why not 1.0?**** Because variable names `a,b` vs `x,y` are different!

###② **Syntax Match** (Code Structure)

****What it does:**** Compares the Abstract Syntax Tree (AST)

****Step-by-step:****

Reference AST nodes:

- FunctionDef (function definition)
- arguments (parameters)
- Return (**return** statement)
- BinOp (binary operation: **+**)

Generated AST nodes:

- FunctionDef ✓
- arguments ✓
- Return ✓
- BinOp ✓

All nodes **match!**

Syntax Match = **4/4 = 1.0** (Perfect!)

****Why 1.0?**** Structure **is** identical even though variable names differ!

###③ **Dataflow Match** (Variable Usage)

****What it does:**** Checks **if** variables are used similarly

****Step-by-step:****

Reference variables: {a, b, add}

Generated variables: {x, y, add}

Common variables: {add}

All variables: {a, b, x, y, add}

Jaccard Similarity = **|intersection| / |union|**

= 1 / 5

= 0.2

Why low? Variable names are completely different!

4 **N-gram Match** (Pattern Matching)

What it does: Checks character/token patterns

Step-by-step:

2-grams:

Reference: [('def', 'add'), ('add', '('), ('(', 'a'), ('a', ')'), ...]

Generated: [('def', 'add'), ('add', '('), ('(', 'x'), ('x', ')'), ...]

Matching 2-grams: ('def', 'add'), ('add', '('), ('(', 'b'), ...

2-gram score ≈ 0.7

3-grams:

Similar process...

3-gram score ≈ 0.6

N-gram Match = (0.7 + 0.6) / 2 = 0.65

⚡ **Final CodeBLEU Score**

CodeBLEU = 0.25 × BLEU + 0.25 × Syntax + 0.25 × Dataflow + 0.25 × N-gram

= 0.25 × 0.65 + 0.25 × 1.0 + 0.25 × 0.2 + 0.25 × 0.65

= 0.1625 + 0.25 + 0.05 + 0.1625

= 0.625

Result: 0.625 → "Fair" !



Why "Fair" and not "Excellent"?

Even though the **code works perfectly**, the score is "Fair" because:

- **Structure is identical** (Syntax = 1.0)
 - **Variable names differ** (Dataflow = 0.2)
 - **Token order differs** (BLEU = 0.65)
-



Better Example - Higher Score:

Reference:

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

Generated (Better Match):

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

Scores:

- BLEU: **1.0** (exact match)
- Syntax: **1.0** (same structure)
- Dataflow: **1.0** (same variables)
- N-gram: **1.0** (same patterns)

CodeBLEU = 1.0 → "Excellent" ✨



Visual Summary:

Metric	What It Checks	Example Score	Why?
BLEU	Word/token similarity	0.65	Different variable names
Syntax	Code structure (AST)	1.0	Same function structure
Dataflow	Variable usage w	0.2	a, b vs x, y

N-gram	Pattern matching	0.65	Some patterns differ
CodeBLU	Weighted average	0.625	Fair quality

Key Takeaway:

CodeBLEU doesn't just check if code *works* - it checks if the generated code is **similar to the reference** in:

1. Text content (BLEU)
2. Structure (Syntax)
3. Logic flow (Dataflow)
4. Patterns (N-gram)