Unsupervised Translation of Programming Languages

Marie-Anne Lachaux, Baptiste Roziere, Lowik Chanussot, Guillaume Lample(2020) | Citation: 72

Group3

Kim Yongtaek (김용택)
Lee Jeongyun (이정윤)
Van Nam Vu

Contents

1. Main Topic

2. How it works

    - Data, Modeling, Evaluation, Results

3. Challenges & Opinion

4. Demo

1. Main Topic

There are various programming languages such as C, C++, Java and Python.

However, the languages are not all compatible according to os or programs, grammar, structure, function, etc. are the same in principle, but there is a difference in the way they are expressed. So far, there has been no translation into a supported language other than rewriting the code by human.

In this project, the goal is to make an Unsupervised Machine Translator which converts programming languages automatically by Deep Learning.

2. How it works

Data

Monolingual open source code from GitHub public dataset

- C++, Java, Python
- over 2.8 million open source GitHub repository
- Valid, Test Set
  - set of 852 parallel functions in 3 languages

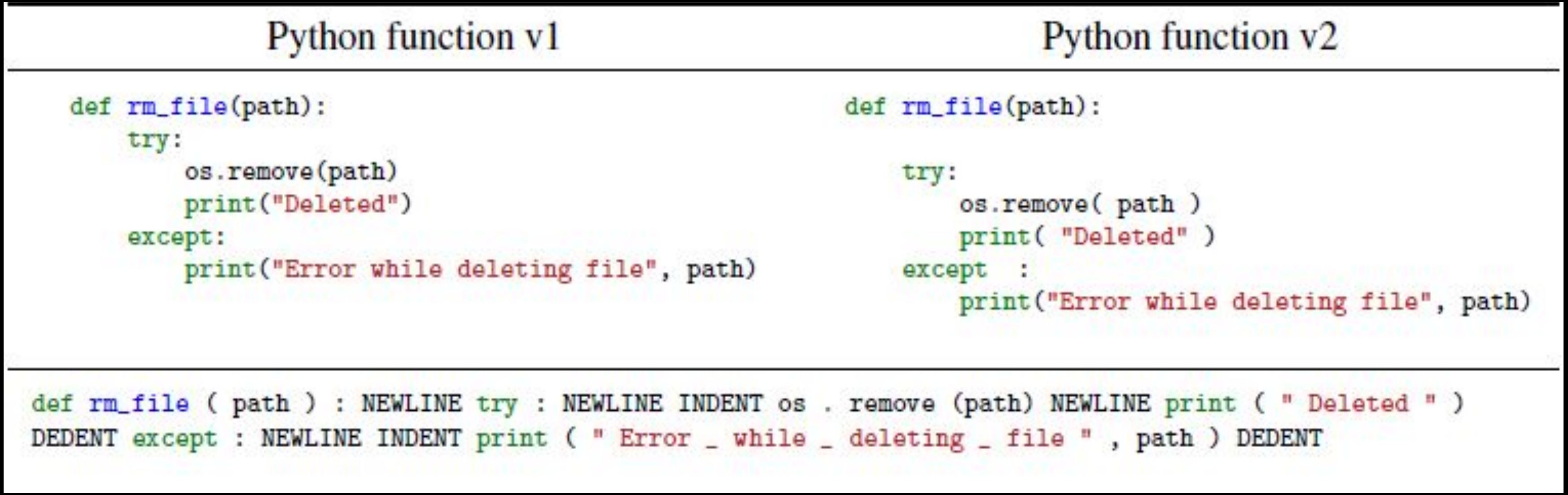
Preprocessing

- Tokenizer: *javalang* (Java), *clang* (C++), *standard library* (Python)
- fastBPE (Byte Pair Encoding)

2. How it works

Data

Python Tokenization



Python function v1

```
def rm_file(path):
    try:
        os.remove(path)
        print("Deleted")
    except:
        print("Error while deleting file", path)
```

Python function v2

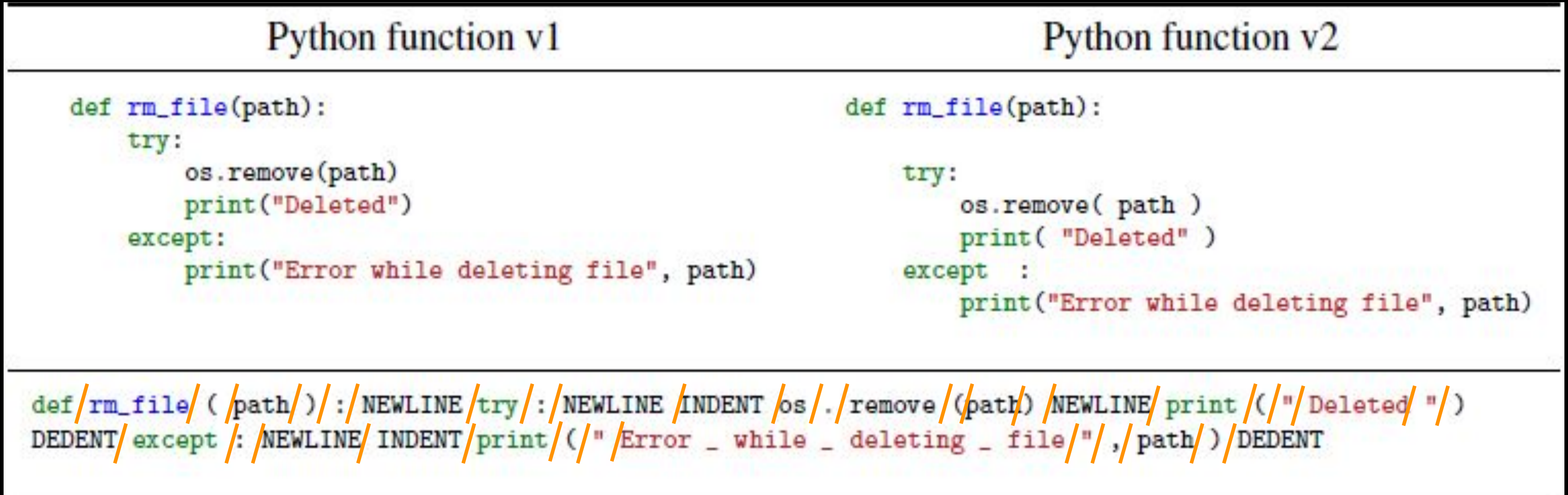
```
def rm_file(path):

    try:
        os.remove( path )
        print( "Deleted" )
    except  :
        print("Error while deleting file", path)
```

```
def rm_file ( path ) : NEWLINE try : NEWLINE INDENT os . remove (path) NEWLINE print ( " Deleted " )
DEDENT except : NEWLINE INDENT print ( " Error _ while _ deleting _ file " , path ) DEDENT
```

2. How it works

Data

Python Tokenization



Python function v1

```
def rm_file(path):
    try:
        os.remove(path)
        print("Deleted")
    except:
        print("Error while deleting file", path)
```

Python function v2

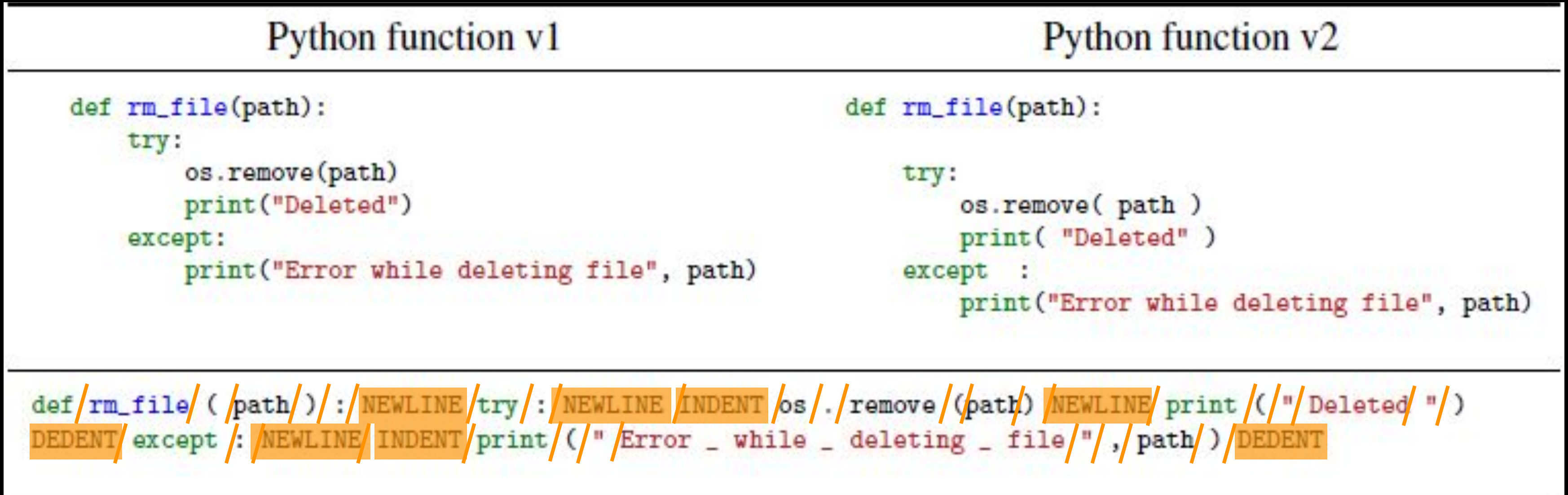
```
def rm_file(path):

    try:
        os.remove( path )
        print( "Deleted" )
    except   :
        print("Error while deleting file", path)
```

```
def rm_file ( path ) : NEWLINE try : NEWLINE INDENT os . remove (path) NEWLINE print ( " Deleted " )
DEDENT except : NEWLINE INDENT print ( " Error _ while _ deleting _ file " , path ) DEDENT
```

2. How it works

Data

Python Tokenization



Python function v1

```
def rm_file(path):
    try:
        os.remove(path)
        print("Deleted")
    except:
        print("Error while deleting file", path)
```

Python function v2

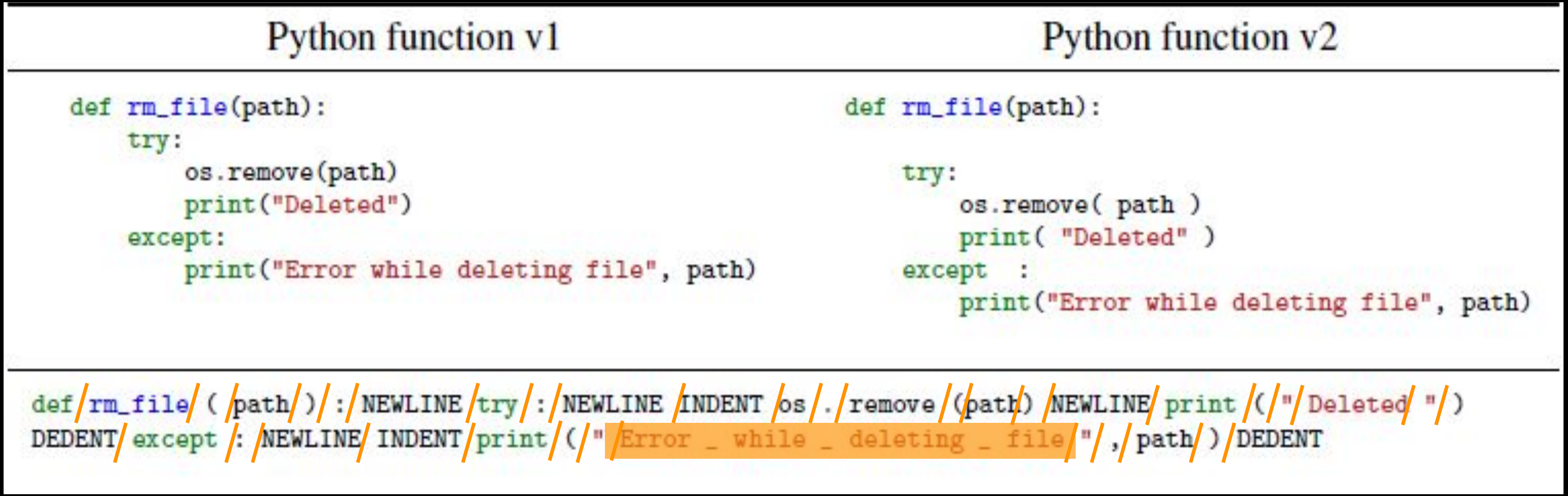
```
def rm_file(path):

    try:
        os.remove( path )
        print( "Deleted" )
    except   :
        print("Error while deleting file", path)
```

```
def / rm_file / ( / path / ) / : / NEWLINE / try / : / NEWLINE / INDENT / os / . / remove / ( / path / ) / NEWLINE / print / ( / " / Deleted / " / )
DEDENT / except / : / NEWLINE / INDENT / print / ( / " / Error _ while _ deleting _ file / " / , / path / ) / DEDENT
```

2. How it works

Data

Python Tokenization



Python function v1

```
def rm_file(path):
    try:
        os.remove(path)
        print("Deleted")
    except:
        print("Error while deleting file", path)
```

Python function v2

```
def rm_file(path):

    try:
        os.remove( path )
        print( "Deleted" )
    except  :
        print("Error while deleting file", path)
```

```
def rm_file ( path ) : NEWLINE try : NEWLINE INDENT os . remove ( path ) NEWLINE print ( " Deleted " )
DEDENT except : NEWLINE INDENT print ( " Error _ while _ deleting _ file " , path ) DEDENT
```

Modeling

Embedding Model: Seq2Seq model with attention

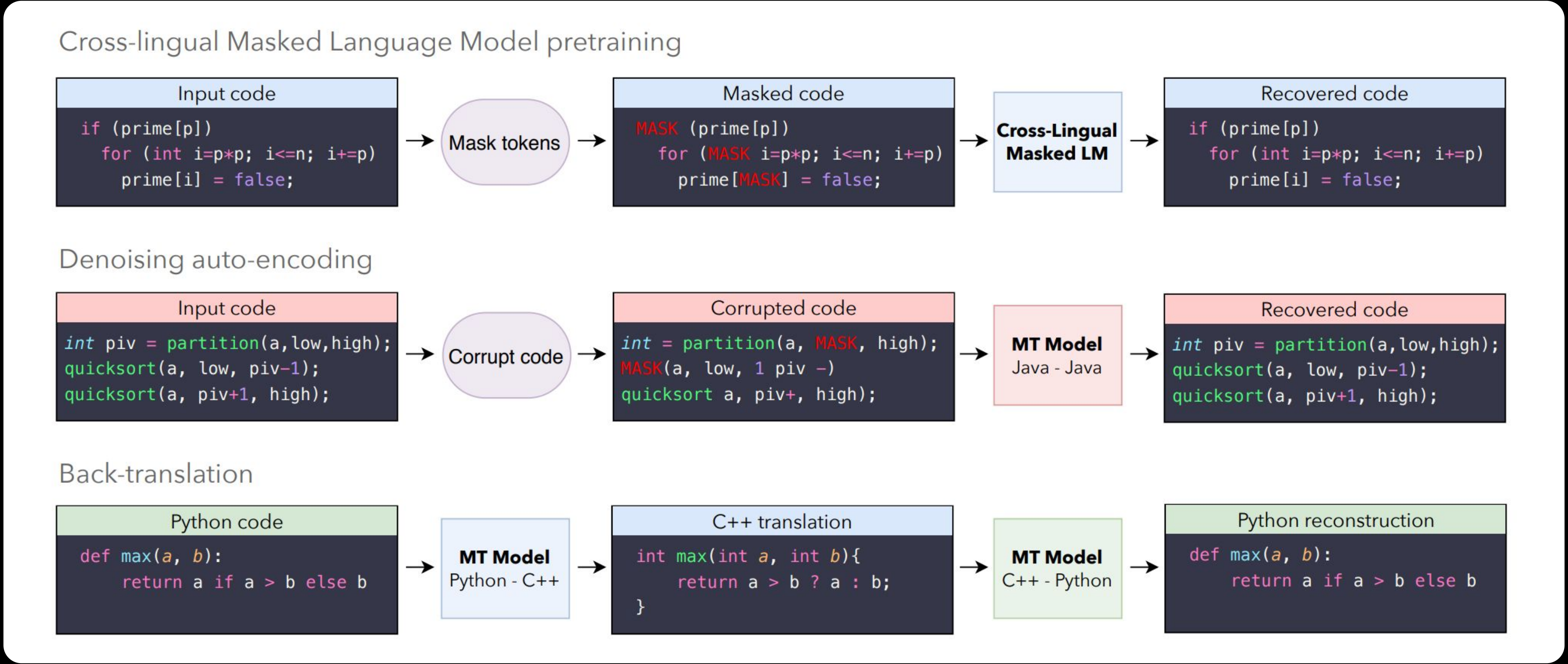
- 6 layers of transformer, 8 attention heads, 1024 dimensions
- single encoder and decoder for all programming languages

Three parts of unsupervised machine translation

- (Pretraining) Cross Programming Language Model pretraining (XLM)
- Denoising auto-encoding (DAE)
- Back-translation

2. How it works

Modeling



Cross-lingual Masked Language Model pretraining

Input code

```
if (prime[p])
  for (int i=p*p; i<=n; i+=p)
    prime[i] = false;
```

Masked code

```
MASK (prime[p])
  for (MASK i=p*p; i<=n; i+=p)
    prime[MASK] = false;
```

Recovered code

```
if (prime[p])
  for (int i=p*p; i<=n; i+=p)
    prime[i] = false;
```

Denoising auto-encoding

Input code

```
int piv = partition(a,low,high);
quicksort(a, low, piv-1);
quicksort(a, piv+1, high);
```

Corrupted code

```
int = partition(a, MASK, high);
MASK(a, low, 1 piv -)
quicksort a, piv+, high);
```

Recovered code

```
int piv = partition(a,low,high);
quicksort(a, low, piv-1);
quicksort(a, piv+1, high);
```

Back-translation

Python code

```
def max(a, b):
    return a if a > b else b
```

C++ translation

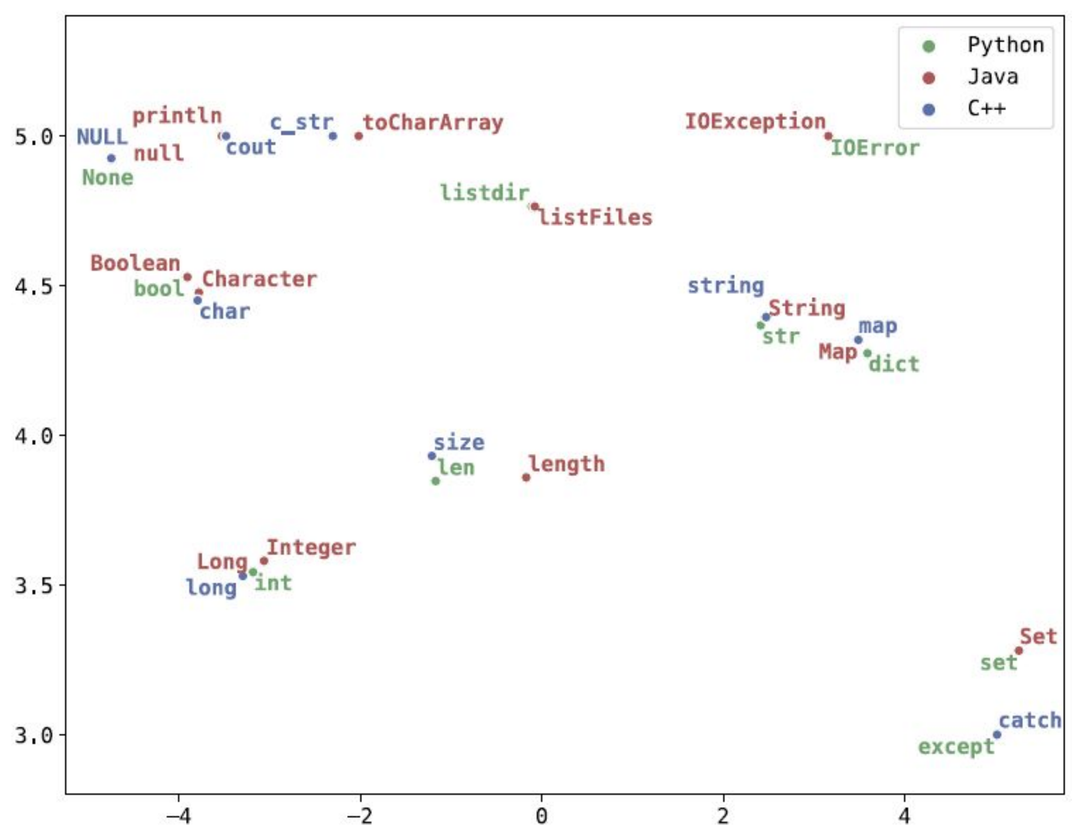
```
int max(int a, int b){
    return a > b ? a : b;
}
```

Python reconstruction

```
def max(a, b):
    return a if a > b else b
```

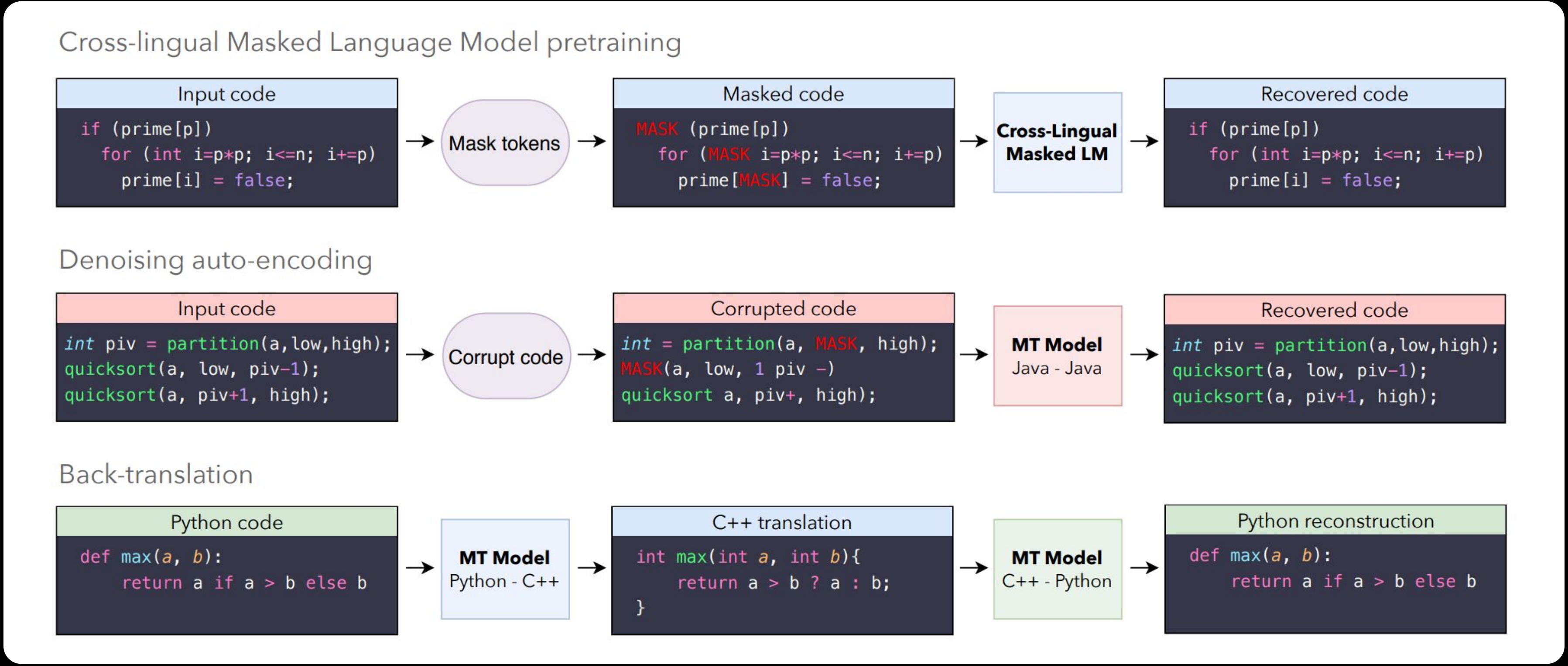
2. How it works

Cross-lingual token embedding space



2. How it works

Modeling



Cross-lingual Masked Language Model pretraining

```
if (prime[p])
  for (int i=p*p; i<=n; i+=p)
    prime[i] = false;
```

```
MASK (prime[p])
  for (MASK i=p*p; i<=n; i+=p)
    prime[MASK] = false;
```

```
if (prime[p])
  for (int i=p*p; i<=n; i+=p)
    prime[i] = false;
```

Denoising auto-encoding

```
int piv = partition(a,low,high);
quicksort(a, low, piv-1);
quicksort(a, piv+1, high);
```

```
int = partition(a, MASK, high);
MASK(a, low, 1 piv –)
quicksort a, piv+, high);
```

```
int piv = partition(a,low,high);
quicksort(a, low, piv-1);
quicksort(a, piv+1, high);
```

Back-translation

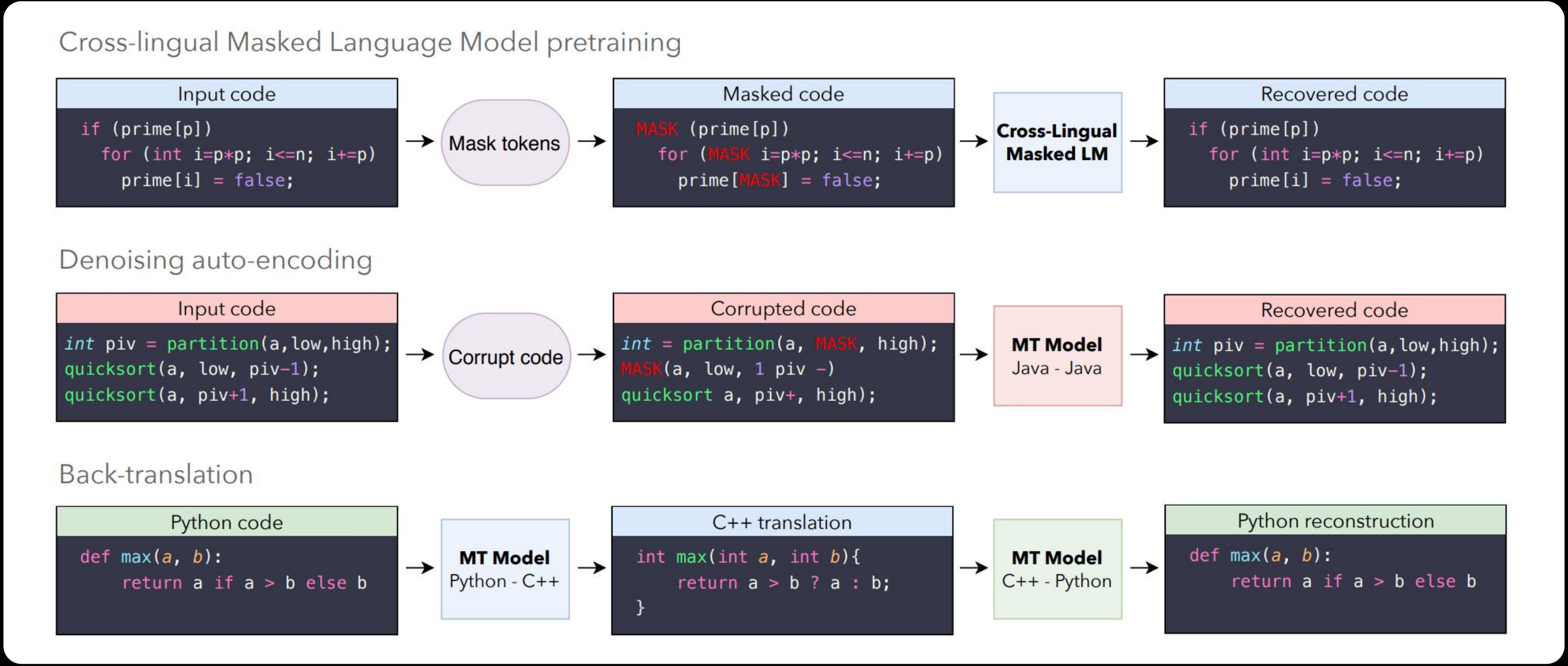
```
def max(a, b):
    return a if a > b else b
```

```
int max(int a, int b){
    return a > b ? a : b;
}
```

```
def max(a, b):
    return a if a > b else b
```

2. How it works

Modeling



Cross-lingual Masked Language Model pretraining

Input code

```
if (prime[p])
  for (int i=p*p; i<=n; i+=p)
    prime[i] = false;
```

→ Mask tokens →

Masked code

```
MASK (prime[p])
  for (MASK i=p*p; i<=n; i+=p)
    prime[MASK] = false;
```

→ **Cross-Lingual Masked LM** →

Recovered code

```
if (prime[p])
  for (int i=p*p; i<=n; i+=p)
    prime[i] = false;
```

Denoising auto-encoding

Input code

```
int piv = partition(a,low,high);
quicksort(a, low, piv-1);
quicksort(a, piv+1, high);
```

→ Corrupt code →

Corrupted code

```
int = partition(a, MASK, high);
MASK(a, low, 1 piv -)
quicksort a, piv+, high);
```

→ **MT Model** Java - Java →

Recovered code

```
int piv = partition(a,low,high);
quicksort(a, low, piv-1);
quicksort(a, piv+1, high);
```

Back-translation

Python code

```
def max(a, b):
    return a if a > b else b
```

→ **MT Model** Python - C++ →

C++ translation

```
int max(int a, int b){
    return a > b ? a : b;
}
```

→ **MT Model** C++ - Python →

Python reconstruction

```
def max(a, b):
    return a if a > b else b
```

target-to-source model

source-to-target model

2. How it works

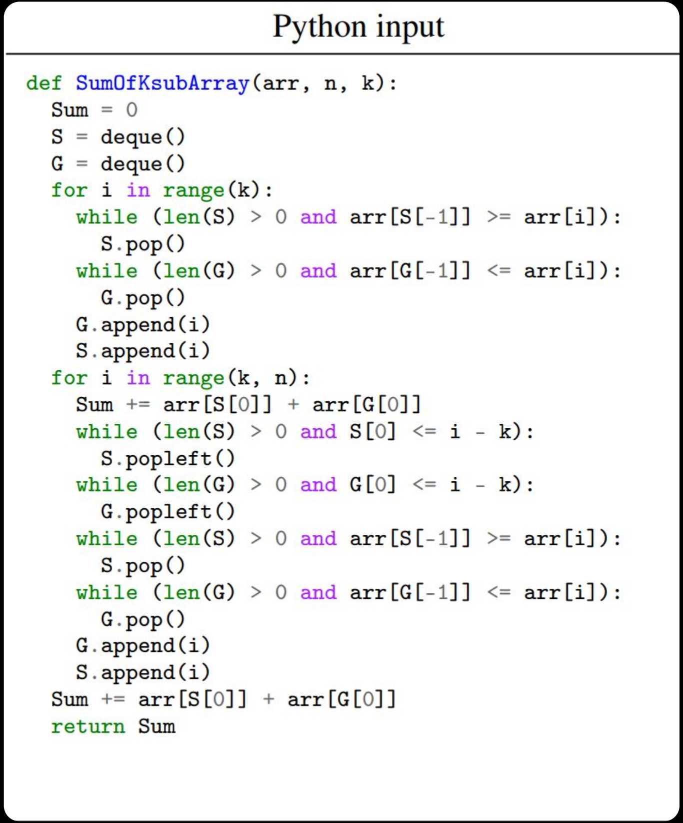
Evaluation

- Set of 852 parallel functions in 3 languages (C++, Java, Python)

  For Validation and Test *from GeeksForGeeks*
- BLEU-Score
- Compute the reference match
- Computational Metric
  - To overcome BLEU limitation
  - Evaluates whether the hypothesis function generates the same outputs as the reference when given the same inputs
- Beam Search Decoding (Greedy Decoding)
  - Beam N : Percentage of functions with at least one correct translation in the beam

2. How it works

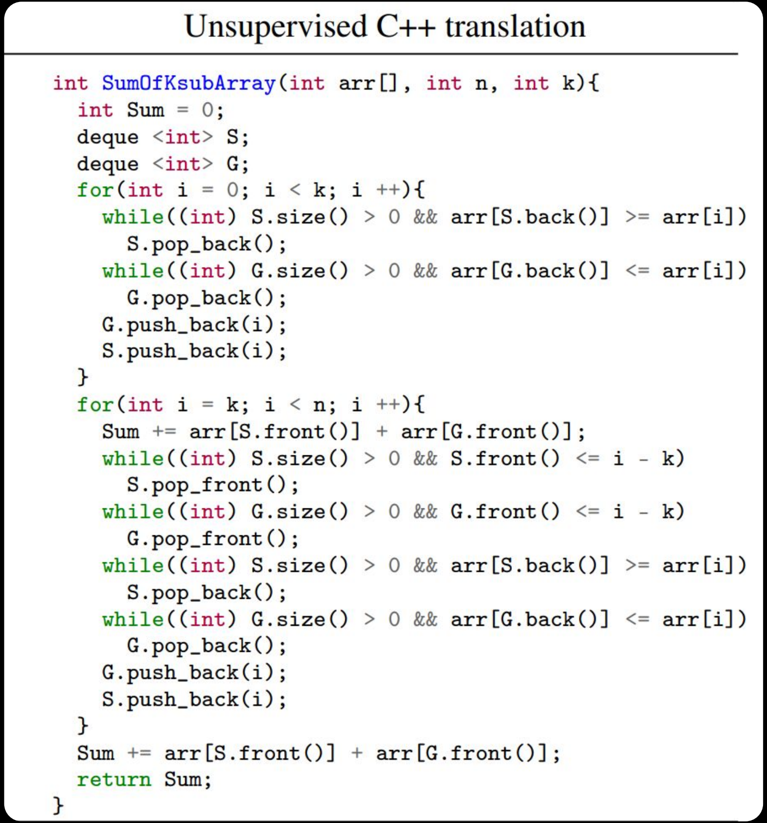
Results



Python input

```python
def SumOfKsubArray(arr, n, k):
  Sum = 0
  S = deque()
  G = deque()
  for i in range(k):
    while (len(S) > 0 and arr[S[-1]] >= arr[i]):
      S.pop()
    while (len(G) > 0 and arr[G[-1]] <= arr[i]):
      G.pop()
    G.append(i)
    S.append(i)
  for i in range(k, n):
    Sum += arr[S[0]] + arr[G[0]]
    while (len(S) > 0 and S[0] <= i - k):
      S.popleft()
    while (len(G) > 0 and G[0] <= i - k):
      G.popleft()
    while (len(S) > 0 and arr[S[-1]] >= arr[i]):
      S.pop()
    while (len(G) > 0 and arr[G[-1]] <= arr[i]):
      G.pop()
    G.append(i)
    S.append(i)
  Sum += arr[S[0]] + arr[G[0]]
  return Sum
```

→

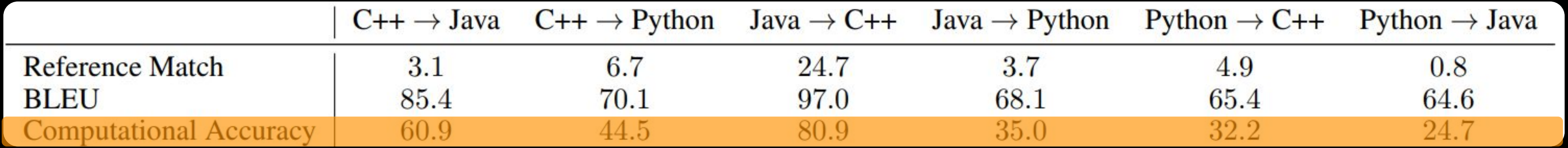


Unsupervised C++ translation

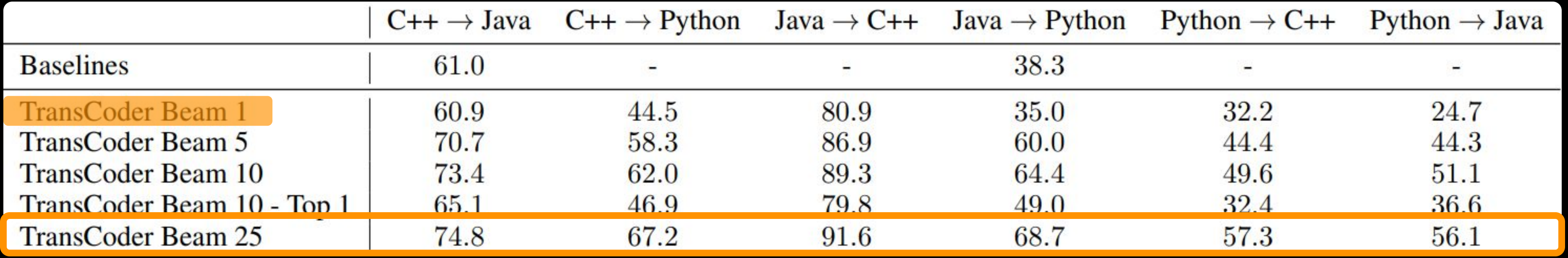
```cpp
int SumOfKsubArray(int arr[], int n, int k){
  int Sum = 0;
  deque <int> S;
  deque <int> G;
  for(int i = 0; i < k; i ++){
    while((int) S.size() > 0 && arr[S.back()] >= arr[i])
      S.pop_back();
    while((int) G.size() > 0 && arr[G.back()] <= arr[i])
      G.pop_back();
    G.push_back(i);
    S.push_back(i);
  }
  for(int i = k; i < n; i ++){
    Sum += arr[S.front()] + arr[G.front()];
    while((int) S.size() > 0 && S.front() <= i - k)
      S.pop_front();
    while((int) G.size() > 0 && G.front() <= i - k)
      G.pop_front();
    while((int) S.size() > 0 && arr[S.back()] >= arr[i])
      S.pop_back();
    while((int) G.size() > 0 && arr[G.back()] <= arr[i])
      G.pop_back();
    G.push_back(i);
    S.push_back(i);
  }
  Sum += arr[S.front()] + arr[G.front()];
  return Sum;
}
```

2. How it works

Results



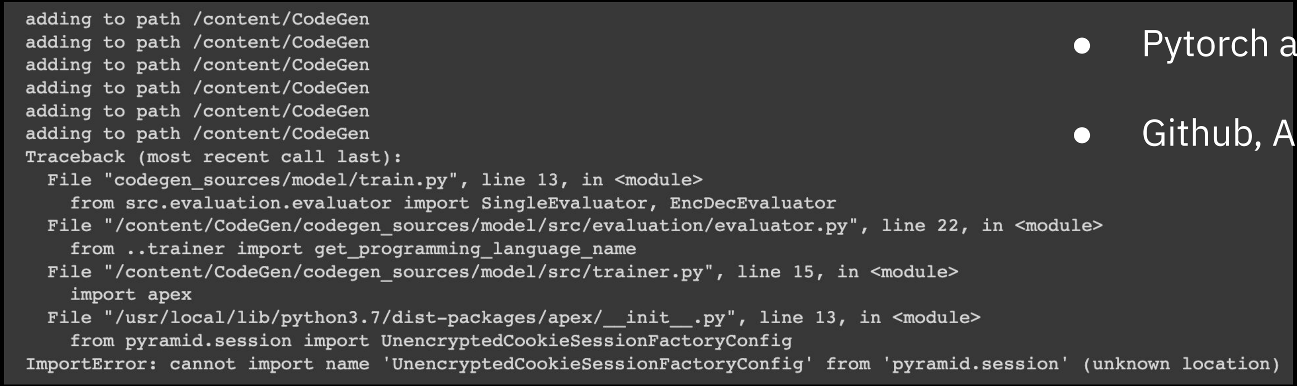
	C++ → Java	C++ → Python	Java → C++	Java → Python	Python → C++	Python → Java
Reference Match	3.1	6.7	24.7	3.7	4.9	0.8
BLEU	85.4	70.1	97.0	68.1	65.4	64.6
Computational Accuracy	60.9	44.5	80.9	35.0	32.2	24.7



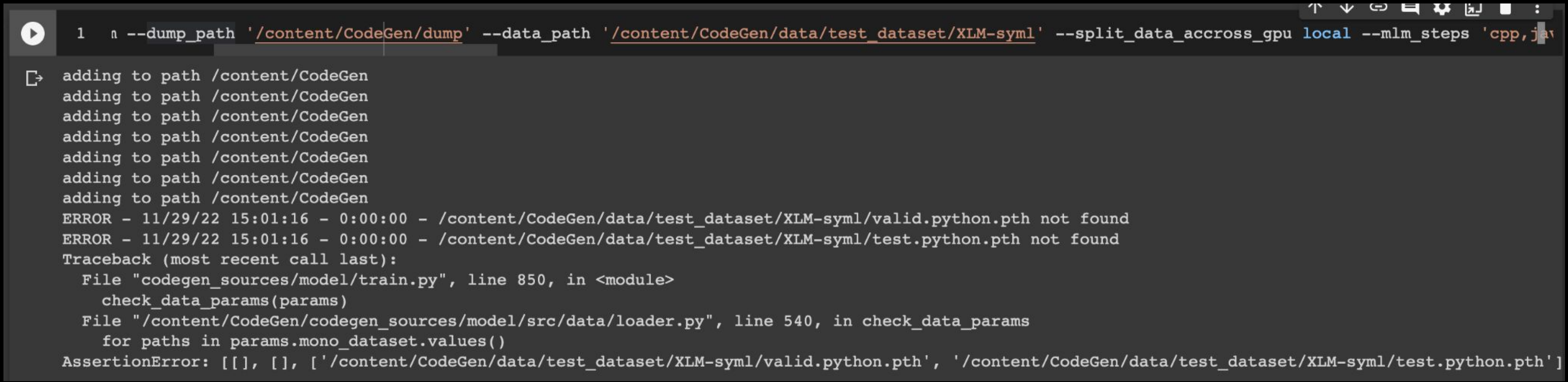
	C++ → Java	C++ → Python	Java → C++	Java → Python	Python → C++	Python → Java
Baselines	61.0	-	-	38.3	-	-
TransCoder Beam 1	60.9	44.5	80.9	35.0	32.2	24.7
TransCoder Beam 5	70.7	58.3	86.9	60.0	44.4	44.3
TransCoder Beam 10	73.4	62.0	89.3	64.4	49.6	51.1
TransCoder Beam 10 - Top 1	65.1	46.9	79.8	49.0	32.4	36.6
TransCoder Beam 25	74.8	67.2	91.6	68.7	57.3	56.1

3. Challenges & Opinion

- Pytorch and CUDA version do not match
- Github, Apex Functions deprecated



```
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
Traceback (most recent call last):
  File "codegen_sources/model/train.py", line 13, in <module>
    from src.evaluation.evaluator import SingleEvaluator, EncDecEvaluator
  File "/content/CodeGen/codegen_sources/model/src/evaluation/evaluator.py", line 22, in <module>
    from ..trainer import get_programming_language_name
  File "/content/CodeGen/codegen_sources/model/src/trainer.py", line 15, in <module>
    import apex
  File "/usr/local/lib/python3.7/dist-packages/apex/__init__.py", line 13, in <module>
    from pyramid.session import UnencryptedCookieSessionFactoryConfig
ImportError: cannot import name 'UnencryptedCookieSessionFactoryConfig' from 'pyramid.session' (unknown location)
```



```
n --dump_path '/content/CodeGen/dump' --data_path '/content/CodeGen/data/test_dataset/XLM-syml' --split_data_accross_gpu local --mlm_steps 'cpp,ja
```

```
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
adding to path /content/CodeGen
ERROR - 11/29/22 15:01:16 - 0:00:00 - /content/CodeGen/data/test_dataset/XLM-syml/valid.python.pth not found
ERROR - 11/29/22 15:01:16 - 0:00:00 - /content/CodeGen/data/test_dataset/XLM-syml/test.python.pth not found
Traceback (most recent call last):
  File "codegen_sources/model/train.py", line 850, in <module>
    check_data_params(params)
  File "/content/CodeGen/codegen_sources/model/src/data/loader.py", line 540, in check_data_params
    for paths in params.mono_dataset.values()
AssertionError: [[], [], ['/content/CodeGen/data/test_dataset/XLM-syml/valid.python.pth', '/content/CodeGen/data/test_dataset/XLM-syml/test.python.pth']]
```

3. Challenges & Opinion

- Automatic translation can make programmers more efficient
  - By allowing them to join various codes from other programmers easily
  - Lower the cost of updating an old codebase written in an obsolete language to a more recent language
  - A powerful tool for programmers for their more innovative projects

- Some mistakes made by the model could be fixed by adding some constraints to the decoder to ensure that the generated functions are syntactically correct, or by using dedicated architectures

- Leveraging the compiler output or other approaches such as iterative error correction could also improve the accuracy of model

4. Demo

Google Colab Links