An Efficient Transformer Alignment Approach for Code Translation using Frozen Models

Siddharth Gandhi, Piyush Khanna, Arjun Choudhry
Language Technologies Institute, Carnegie Mellon University

Carnegie Mellon University School of Computer Science

Carnegie Mellon University Language Technologies Institute

Motivation

Code translation is used for the following:

- **Omnipresent languages** like Java to new and **task-specific languages** like DART.

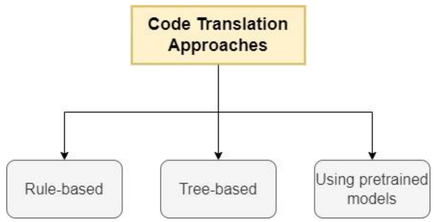
```cpp
// C++
int nthTerm ( int n ) {
  return ( n*n ) + ( n*n*n ) ;
}
```

```rust
// Rust
fn nth_term ( n : i32 ) -> i32 {
    ( n*n ) + ( n*n*n )
}
```

```java
// Java
static int divisorSum(int n){
  int sum = 0 ;
  for(int i=1; i<=n; ++i )
    sum += ( n / i ) * i;
  return sum ;
}
```

```go
// Go
func divisorSum ( n int ) int {
 sum := 0
 for i := 1 ; i <= n ; i ++ {
   sum += ( n / i ) * i
 }
 return sum
}
```

- Migrating code from **legacy languages** like COBOL to **modern ones** like Java.
- Translating code into **application-specific packages** (like Keras to PyTorch).



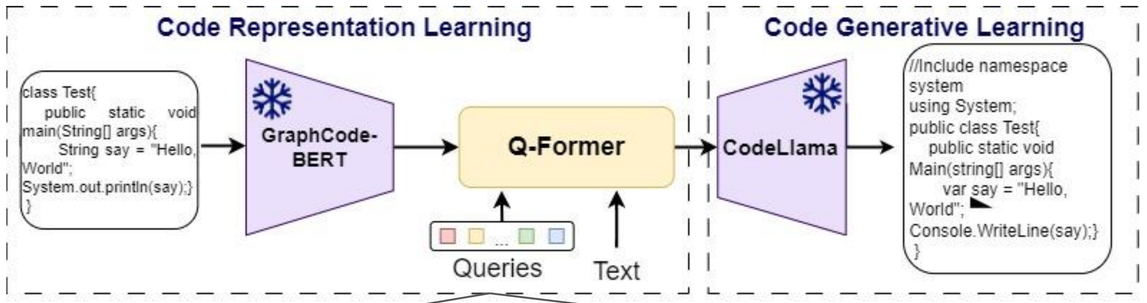
Our Work

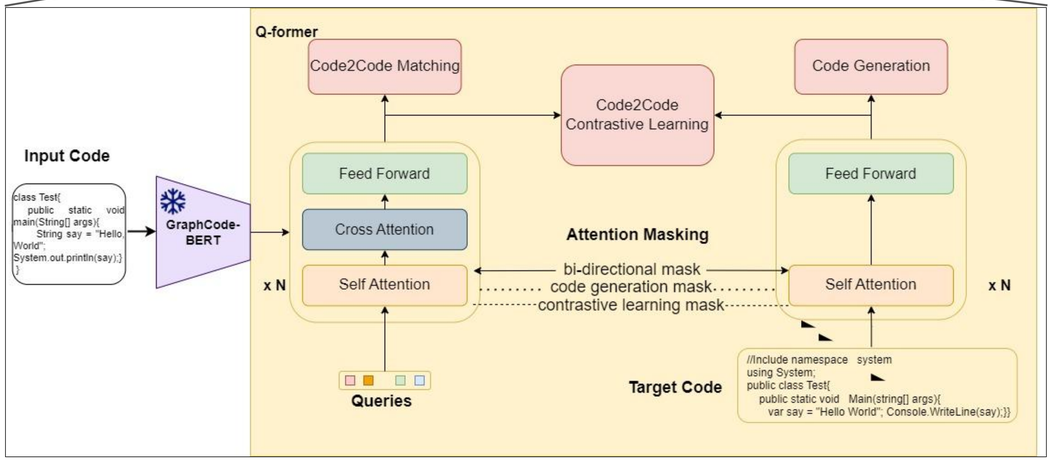
We propose a code translation model built using **frozen code transformer models** aligned using a **Querying Transformer** for efficient training, based on the BLIP-2 image-to-text model.

Drawbacks of Current Models

- Either **no scope for zero-shot translation**, or **translations are random**.
- **Computationally expensive**, since either require **large-scale pretraining** or **finetuning the entire model**, which can have parameters in the range of billions of parameters.
- Relies entirely on **available paired data** for learning to do translation, and **overfits easily** to the training data style.

Methodology

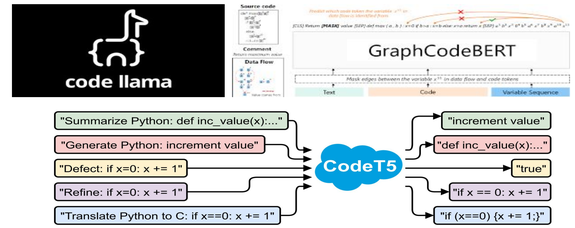




Training happens in 2 steps:

- **Representation Learning:** Use frozen pretrained encoder and train the Q-Former to learn query embeddings that capture salient code representations, informative of input text. The Q-Former is trained using three objectives:
  - **contrastive loss** to align code representations from input programming language with code representations from output programming language by maximizing their mutual information
  - **code generation loss** to train the Q-Former to generate code conditioned on input code.
  - **Code2Code matching loss** for finer grained alignment between code representations across programming languages.
- **Generative Learning:** Connect the frozen encoder and Q-Former, with the frozen decoder. The query embeddings serve as soft prompts to guide the decoder's code generation. Trained using the causal language modeling loss.

Transformers Experimented With



Preliminary Results for Java To C#

Method	BLEU	Weighted n-gram Match	Syntax Match	Dataflow Match	Code BLEU
GraphCode BERT	81.59	82.23	88.79	87.91	85.13
Our Initial Model	14.56	17.75	25.80	14.50	18.14

We observe substantially lower performance for our model, compared to the baseline. This can be attributed to even the best code decoders being poor code translators. We evaluated simple code translation for CodeLlama-7B. An example is as follows:

Java: public String toString() {return getClass().getName() + " [" +_value +"]";}

C#: public override String ToString(){StringBuilder sb = new StringBuilder(64); sb.Append(GetType().Name).Append(" ["); sb.Append(value); sb.Append("]");return sb.ToString();} """ + f" Java: {java_code} \n C#:"

Future Goals

- More hyperparameter tuning to improve performance.
- Train for other tasks, including code refinement and code completion, simultaneously while training for translation to train a more general model and potentially enabling multiple tasks across programming languages in zero-shot settings.
- Curate a large dataset for code translation from CodeForces to better train the Q-Former.
- Experiment with more encoders.