Deep Code Comment Generation

Xing Hu[1] , Ge Li[1], Xin Xia[2], David Lo[3], Zhi Jin[1]

[1] Key Laboratory of High Confidence Software Technologies (PKU), China
[2] Faculty of Information Technology, Monash University, Australia
[3] School of Information Systems, Singapore Management University, Singapore

ICPC 2018
May 28 Gothenburg

Why Code Summarization?

- **Code Comprehension**
  - Comments are often missed, mismatch, outdated, …
- **Summarization**
  - Aims to obtain a reductive transformation from a source text to a summary text through different techniques.

In software development and maintenance, developers spend around **59%** of their time on program comprehension activities*

Xia X, Bao L, Lo D, et al. Measuring program comprehension: A large-scale field study with professionals[J]. IEEE Transactions on Software Engineering, 2017.

Existing Approaches

- **Information Retrieval Approaches**
  - Extract natural descriptions from software artifacts, e.g., bug report, Stack Overflow…
  - Extract keywords from source code

Limitations:

- Heavily rely on whether similar code snippets can be retrieved and how similar the snippets are.
- Fail to extract accurate keywords when identifiers and methods are poorly named.

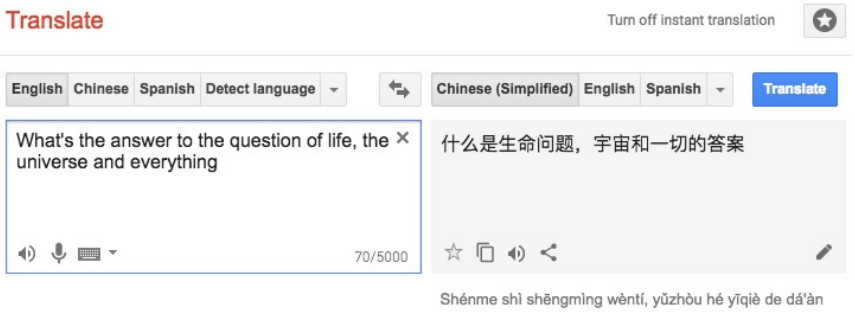
Code Summarization VS NMT

Code Summarization: Bridges gap between Source Code and Natural Language

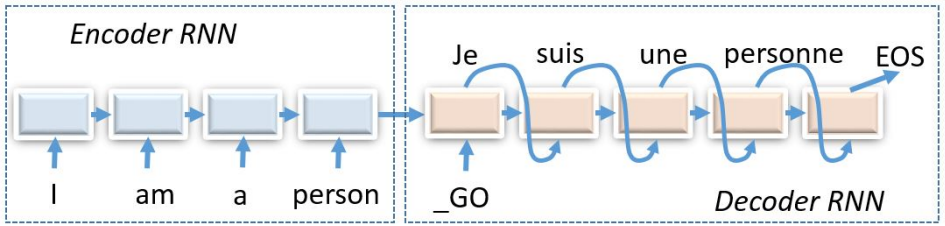
```
public static void sort(Comparable[] a){
    int n=a.length;
    for (int i=1; i < n; i++){
        Comparable v=a[i];
        int lo=0, hi=i;
        while (lo < hi) { ... }
        ...
    }
    assert isSorted(a);
}
```

Comment:
Rearranges the array in ascending order,using the natural order.

NMT: aims to translate from one NL language (e.g., English) to another NL language (e.g., Chinese)



NMT Model:



Background: Neural Machine Translation

- **A deep learning model for the sequence-to-sequence model**

  - Encoder: An RNN that encodes a sequence of words (code)

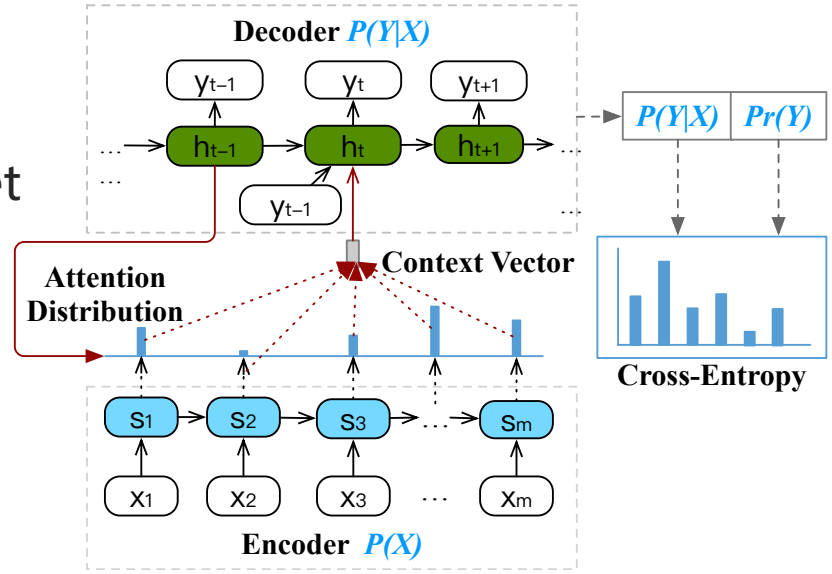
$$s_t = f(x_t, s_{t-1})$$

  - Attention: Selects the important parts from the input sequence for each target word.

$$c_i = \sum_j^m \alpha_{ij} s_j$$

  - Decoder: An RNN that sequentially generates a sequence of words (comments)

$$p(y_i|y_1, \dots, y_{i-1}, X) = g(y_{i-1}, h_i, c_i)$$



Intuitive Overview

```
public static void sort(Comparable[]
a){
    int n=a.length;
    for (int i=1; i < n; i++){
        Comparable v=a[i];
        int lo=0, hi=i;
        while (lo < hi) { ... }
        ...
    }
    assert isSorted(a);
}
```

Java method without comment

Comment generation phase

Program Parser (Tokens)

Comment Tokens

Sorts the array in ascending order,using the natural order.

Code comment

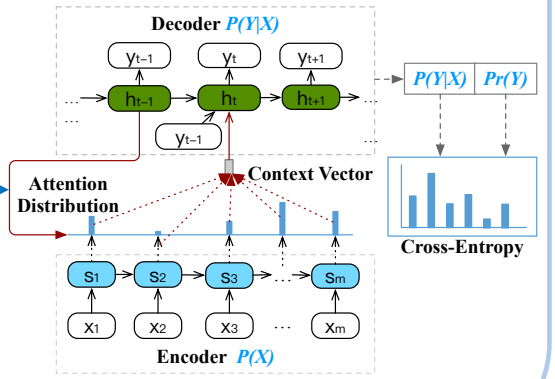
NMT Model

Training phase

Java Methods

Program Parser (Tokens)

Comment Parser (Tokens)



Source Code VS NL

Program Language (PL)

Natural Language (NL)

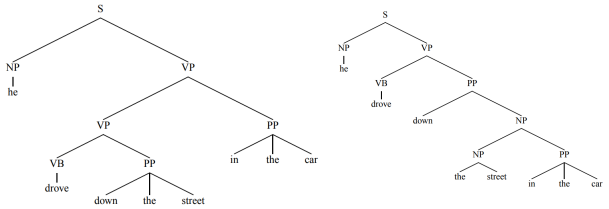
Source Code VS NL

Structural language

Program Language (PL)	Natural Language (NL)
Strong Unambiguous	**Weak Textual ambiguity**

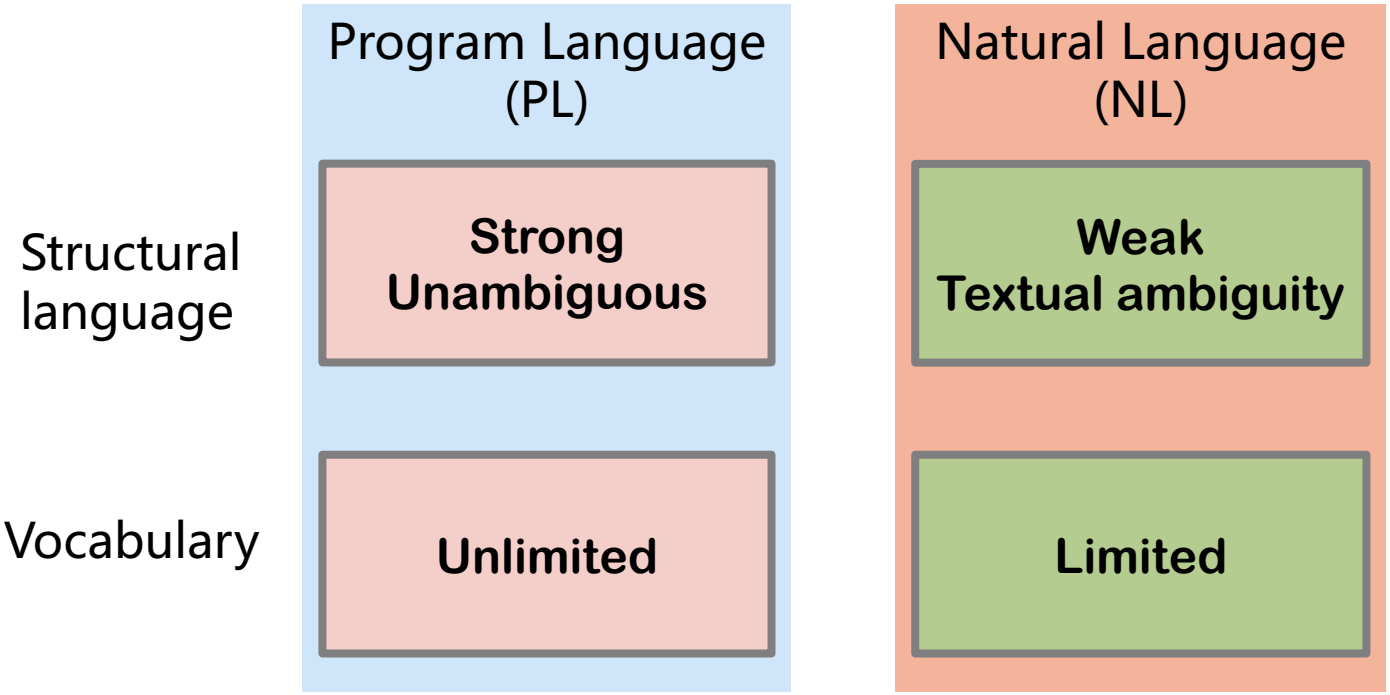
Source Code

Abstract Syntax Tree



he drove down the street in the car

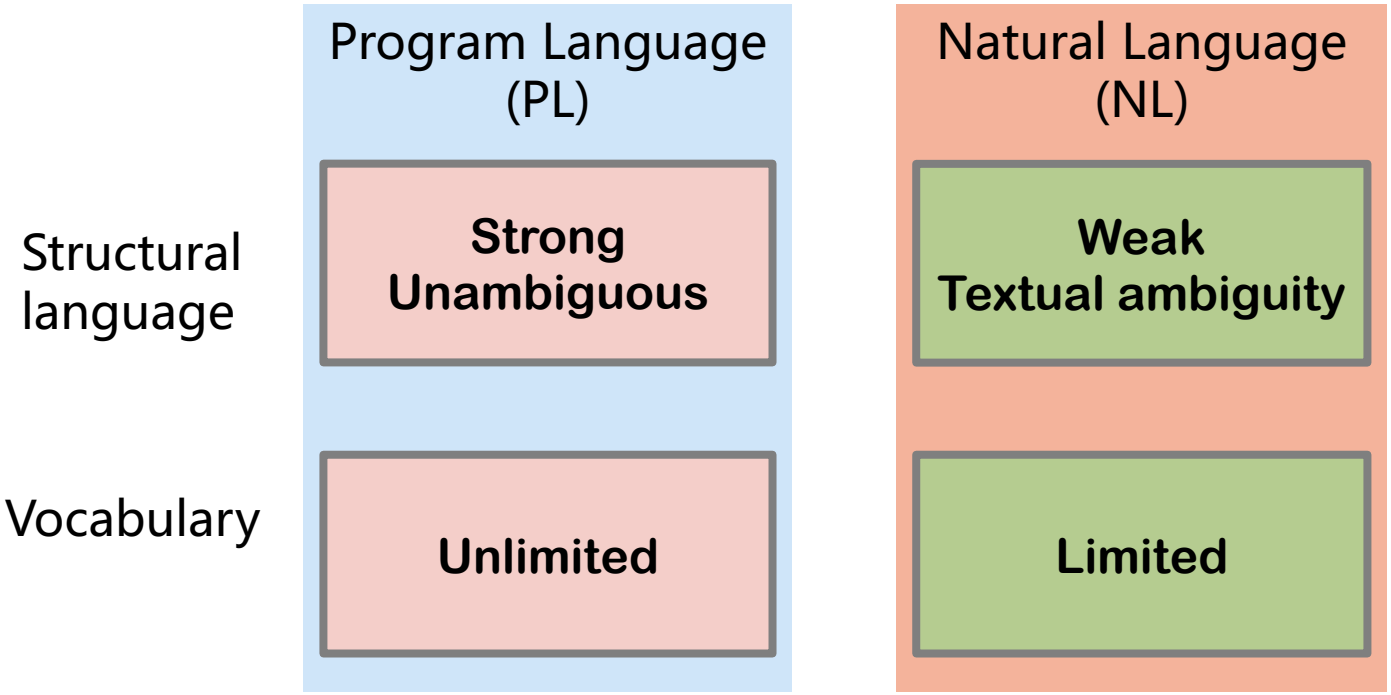
Source Code VS NL



#All Tokens	#All Identifiers	# Unique Tokens	#Unique Identifiers
44,378,497	13,779,297	794,711	794,621

- If the vocabulary size is 30,000, 95% identifiers are <UNK>
- If we want the occurrences of ⟨UNK⟩ tokens to be as few as possible, the vocabulary size will increase a lot.

Source Code VS NL



Challenges:

- Utilize rich and unambiguous structure information
- Reduce unknown words in source code

Key Idea

```
public static void sort(Comparable[]
a){
    int n=a.length;
    for (int i=1; i < n; i++){
        Comparable v=a[i];
        int lo=0, hi=i;
        while (lo < hi) { ... }
        ...
    }
    assert isSorted(a);
}
```

Java method without comment

Comment generation phase

Program Parser (Tokens)

Comment Tokens

Sorts the array in ascending order,using the natural order.

Code comment

NMT Model

Training phase

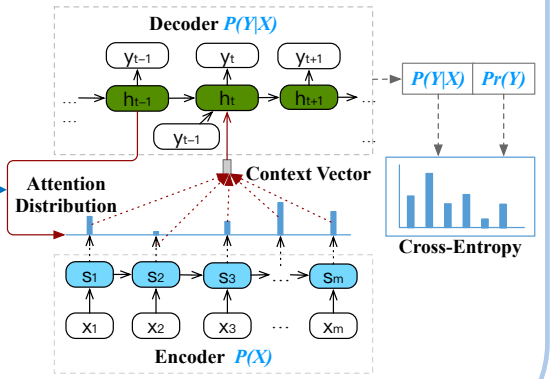
Java Methods

Program Parser

How

UNKs-> Type

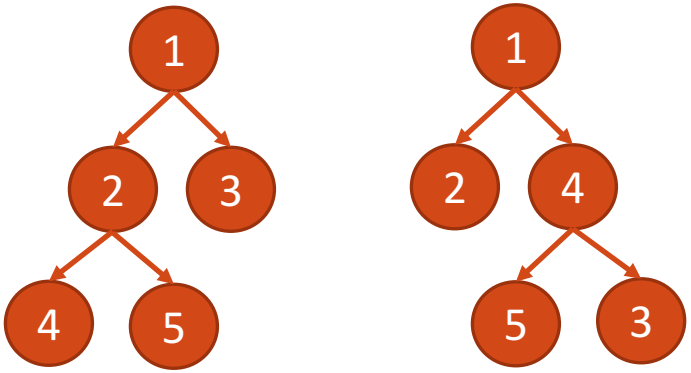
Comment Parser (Tokens)



github SOCIAL CODING

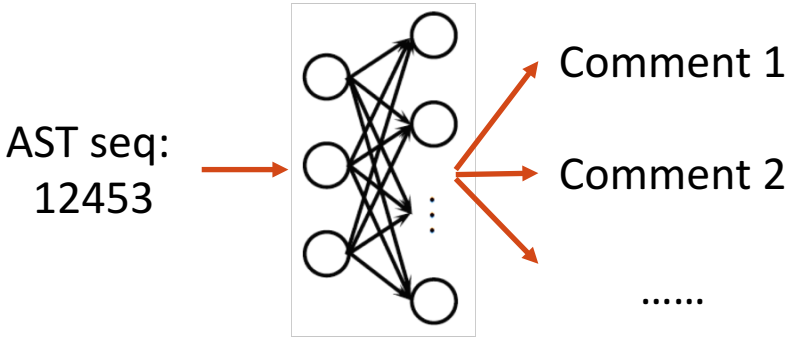
Abstract Syntax Tree with SBT traversal

- Traditional traversal methods, e.g., pre-order
  - Lossy since the original ASTs cannot unambiguously be reconstructed back from them.



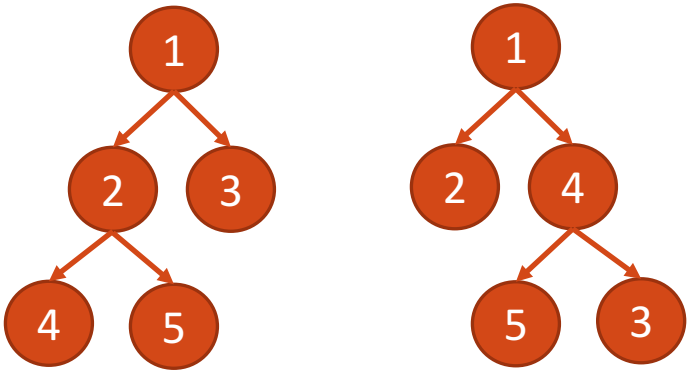
Pre-order： 12453

- Different trees (i.e., different Java methods) are mapped into the same AST sequence.
- DNN is confused if there are multiple labels (in our setting, comments) given to a specific input.



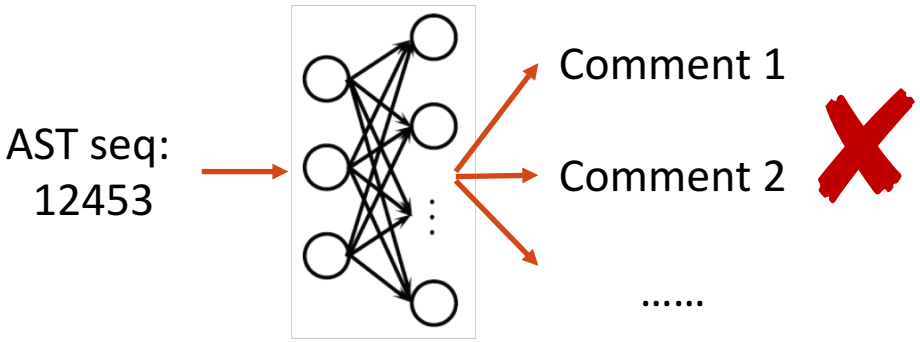
Abstract Syntax Tree with SBT traversal

- Traditional traversal methods, e.g., pre-order
  - Lossy since the original ASTs cannot unambiguously be reconstructed back from them.



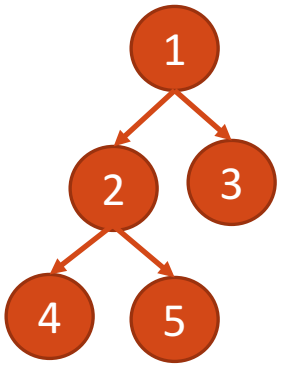
Pre-order： 12453

- Different trees (i.e., different Java methods) are mapped into the same AST sequence.
- DNN is confused if there are multiple labels (in our setting, comments) given to a specific input.



Abstract Syntax Tree with SBT traversal

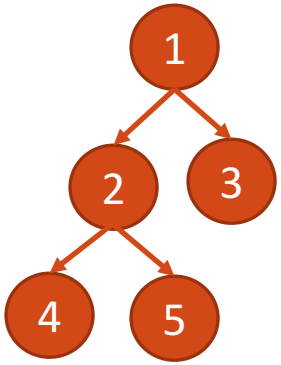
- **Structure-based Traversal**
  - Use brackets keeping tree structures



(1)**1**

(1(2)**2**(3)**3**)**1**

(1(2(4)**4**(5)**5**)**2**(3)**3**)**1**



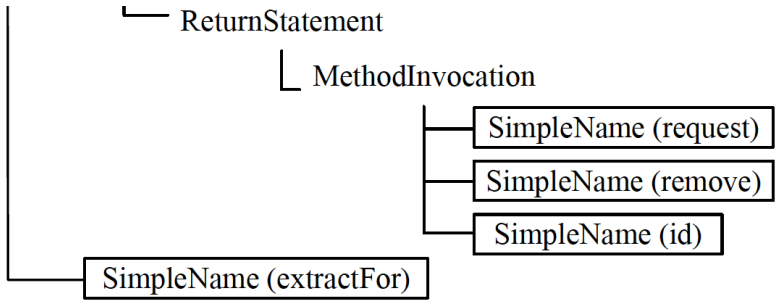
(1(2(4)**4**(5)**5**)**2**(3)**3**)**1**

(2(4)**4**(5)**5**)**2**(3)**3**

(4)**4**(5)**5**

Out-of-Vocabulary tokens

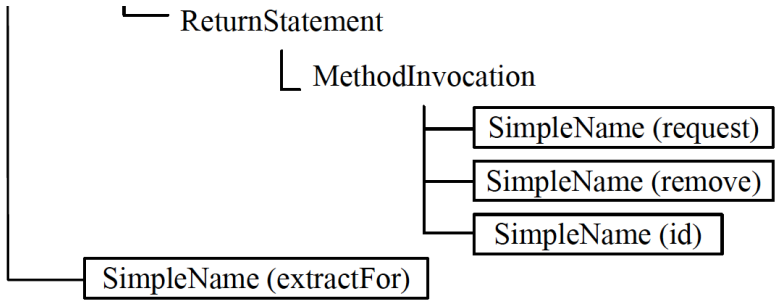
- **Node representation**
  - Non-terminal nodes: type
  - Terminal nodes: type_value
- **Vocabulary**
  - Top 30,000 tokens, including (non-)terminal type, type_value, brackets.
- OOV type_value pairs are replaced by their type.

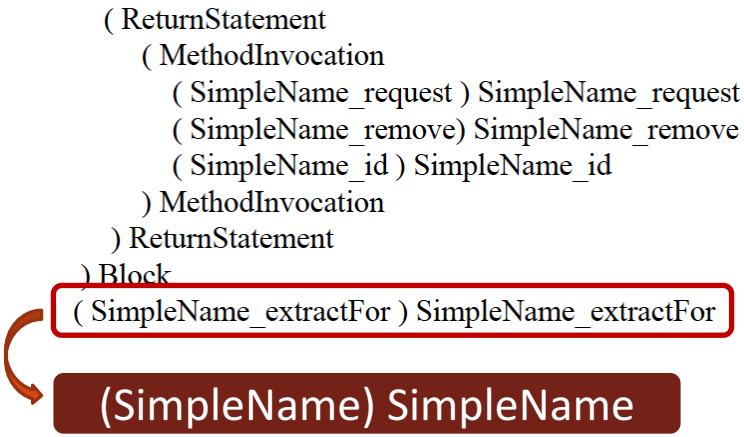


(ReturnStatement
(MethodInvocation
(SimpleName_request) SimpleName_request
(SimpleName_remove) SimpleName_remove
(SimpleName_id) SimpleName_id
) MethodInvocation
) ReturnStatement
) Block
(SimpleName_extractFor) SimpleName_extractFor

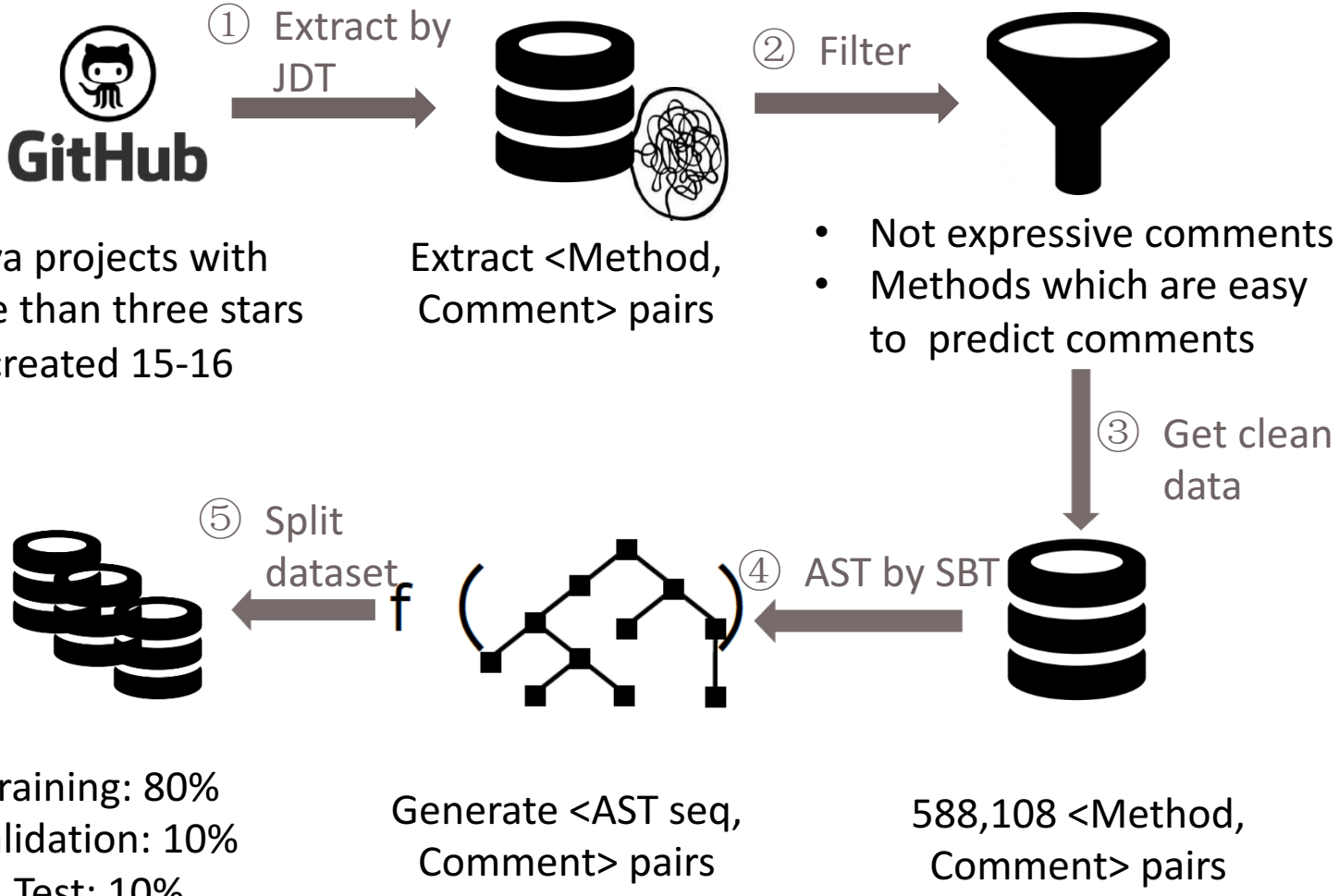
Out-of-Vocabulary tokens

- **Node representation**
  - Non-terminal nodes: type
  - Terminal nodes: type_value
- **Vocabulary**
  - Top 30,000 tokens, including (non-)terminal type, type_value, brackets.
- OOV type_value pairs are replaced by their type.





Experiment: Dataset



Results

- **RQ1: How effective is DeepCom compared with the state-of-the-art baseline?**
- **RQ2: How effective is DeepCom to source code and comments of varying lengths?**

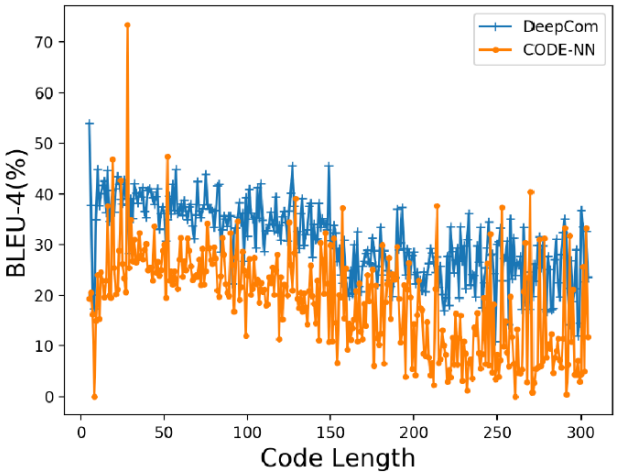
Results

- **RQ1: DeepCom VS baselines**
  - Java methods
  - Baseline: CODE-NN
  - Accuracy Metric: BLEU

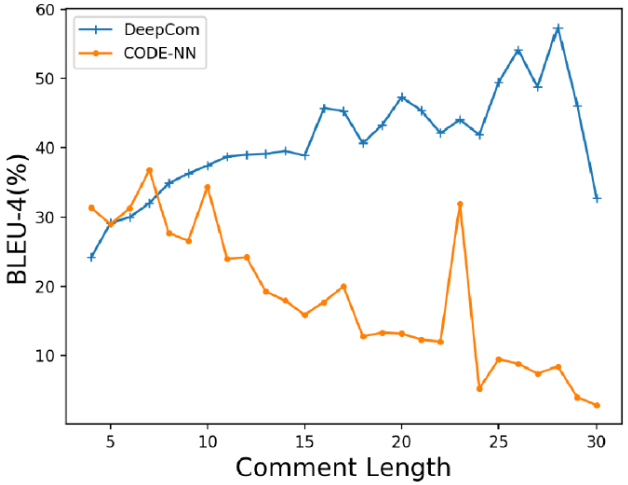
Evaluation results on Java methods	
Approaches	**BLEU-4 score (%)**
CODE-NN	25.30
Seq2Seq	34.87
Attention-based Seq2Seq	35.50
DeepCom (Pre-order)	36.01
DeepCom (SBT)	**38.17**

Results

- **RQ2: Accuracy under different lengths of source code and comments**



(a) BLEU-4 scores for different code lengths



(b) BLEU-4 scores for different comment lengths

- For most code lengths, the average scores of DeepCom improve those of CODE-NN by about 10%.
- For comments of different lengths, the accuracy of CODE-NN decreases sharply and DeepCom still performs better.

Discussion

- Exactly correct comments
- Algorithm implementations
- Cases when generated comments are better than human-written ones
- API invocations intensive Java methods
- Unknown words in generated comments

Discussion

- **Exactly correct comments**
  - Clear business logic
  - Universal code conventions

```
public static byte[] bitmapToByte(Bitmap b){
    ByteArrayOutputStream o = new ByteArrayOutputStream();
    b.compress(Bitmap.CompressFormat.PNG,100,o);
    return o.toByteArray();
}
```

DeepCom: convert Bitmap to byte array
Human-Written: convert Bitmap to byte array

```
private static void addDefaultProfile(SpringApplication app,
                SimpleCommandLinePropertySource source){
    if(!source.containsProperty("spring.profiles.active")
                &&!System.getenv().containsKey("SPRING_PR
                                                OFILES_ACTIVE")){
        app.setAdditionalProfiles
                (Constants.SPRING_PROFILE_DEVELOPMENT);
    }
}
```

DeepCom: If no profile has been configured , set by default the “dev” profile.
Human-Written: If no profile has been configured , set by default the “dev” profile.

Discussion

- **Algorithm implementations**
  - Java methods usually use similar structures to implement the same algorithm function.
  - Capture the correct functionality

```
public static void sort(Comparable[] a){
    int n=a.length;
    for (int i=1; i < n; i++){
        Comparable v=a[i];
        int lo=0, hi=i;
        while (lo < hi) { ... }
        ...
    }
    assert isSorted(a);
}
```

DeepCom: Sorts the array in ascending order, using the natural order.
Human-written: Rearranges the array in ascending order, using the natural order.

Discussion

- **Cases when generated comments are better than human-written ones**
  - Determine something true or not.
  - Developers write interrogative sentences sometimes.

```
public boolean isEmpty(){
      return root == null;
}
```

DeepCom: Returns true if the symbol is empty.
Human-written: Is this symbol table empty?

```
public boolean contains(int key){
      return rank(key) != -1;
}
```

DeepCom: Checks whether the given object is contained within the given set.
Human-written: Is the key in this set of integers?

Discussion

- **API invocations intensive Java methods**

```
public static byte[] bitmapToByte(Bitmap b){
    ByteArrayOutputStream o = new ByteArrayOutputStream();
    b.compress(Bitmap.CompressFormat.PNG,100,o);
    return o.toByteArray();
}
```

DeepCom: convert Bitmap to byte array
Human-Written: convert Bitmap to byte array

```
protected void createItemsLayout(){
  if (mItemsLayout == null){
     mItemsLayout=new LinearLayout(getContext());
     mItemsLayout.setOrientation(LinearLayout.VERTICAL);
  }
}
```

DeepCom: Creates item layouts if any parameters
Human-written: Creates item layout if necessary

```
public void unlisten(String pattern){
    UtilListener listener=listeners.get(pattern);
    if(listener!=null){
       listener.destroy();
       listeners.remove(pattern);
    }else{
      client.onError(Topic.RECORD,
                     Event.NOT_LISTENING,pattern);
  }
}
```

DeepCom: It can be called when the produ
only or refresh has ended.
Human-written: Removes a listener that was previously registered with listenFor-Subscriptions.

Dicussion

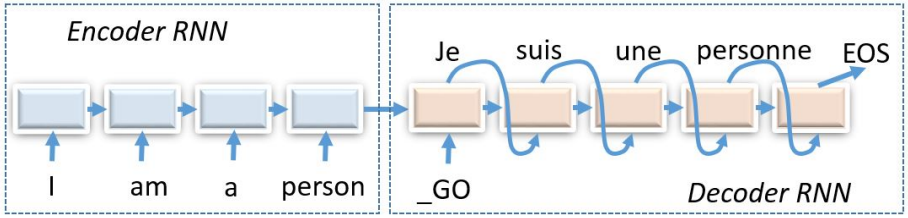
- **Unknown words in generated comments**
  - Not good at learning the method or identifiers names occurred in comments.

```
public FactoryConfigurationError(Exception e){
   super(e.toString());
   this.exception=e;
}
```

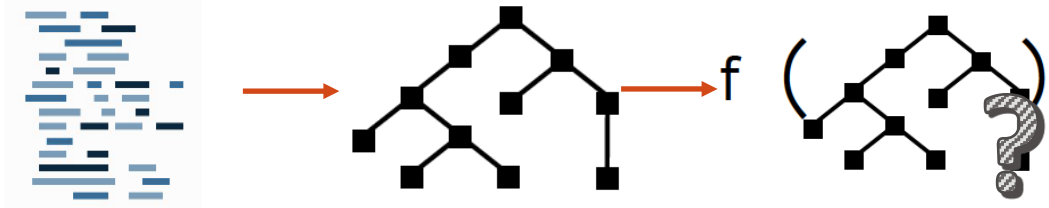
DeepCom: Create a new ⟨UNK⟩ with a given Exception base cause of the error.
Human-written: Create a new FactoryConfigurationError with a given Exception base cause of the error.

Conclusion

DeepCom: formulate code comments generation task as a machine translation task.



SBT: keep the AST sequences unambiguous



OOV: domain-specific method to deal with OOV

```
private static void UNK(SpringApplication app, UNK source){
    if(!source.UNK("spring.profiles.active")
            &&!System.getenv().containsKey(){
        app.UNK (Constants.UNK);
    }
}
```

SimpleName

Future Work: Domain-specific customizations

Q&A

Thanks

BLEU: the Automatic Metric

- **A popular metric for measuring the similarity between two sentences in NMT.**
- **Range from 0 to 1**

DeepCom VS CODE-NN

- **CODE-NN**
  - Use an RNN decoder with the attention to generate summaries by integrating the toke embeddings of source code instead of building language models for source code.
- **DeepCom**
  - Building language model for both source code and comments