Vector Embedding of Code

Code Repository Mining (CRM2020)

Lando Löper
Software Architecture Group
21.07.2020

Context

New Project Old Code

- New project based on an older project
- Poorly maintained legacy code
- Developers are not available anymore

Problem

Deciphering Is Arduous

- What? > How?
- Similar syntax does not imply similar behaviour
- Direct mapping from code to text is non trivial

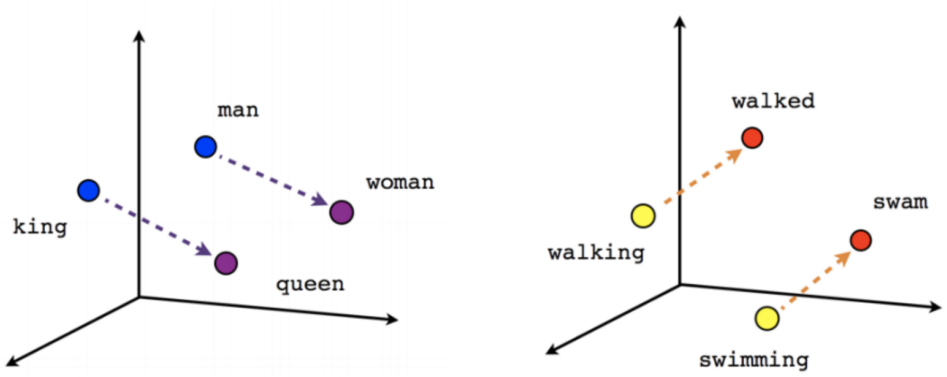
Approach

Leverage concepts of NLP

- Vector embeddings
- Seq2Seq learning
- Example: Machine translation

Approach

Vector embeddings



Male-Female

Verb tense

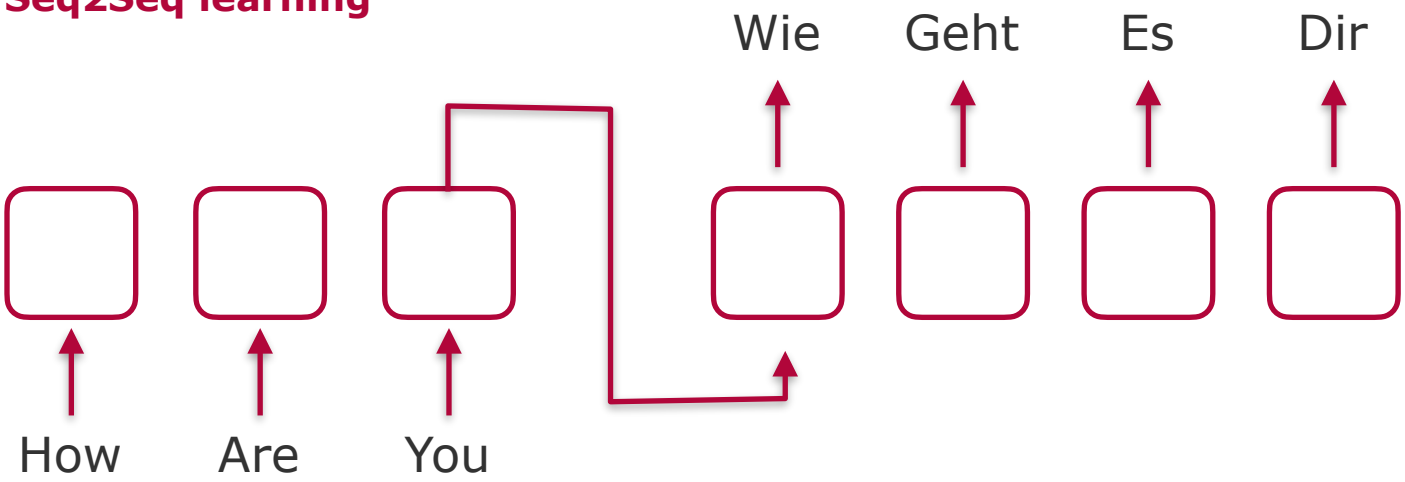
- Technique of mapping semantic meaning into a geometric space
- Distance between two vectors captures a semantic relationship

Vector Embedding of Code

Lando Löper, Code Repository Mining, 21.07.2020

Approach

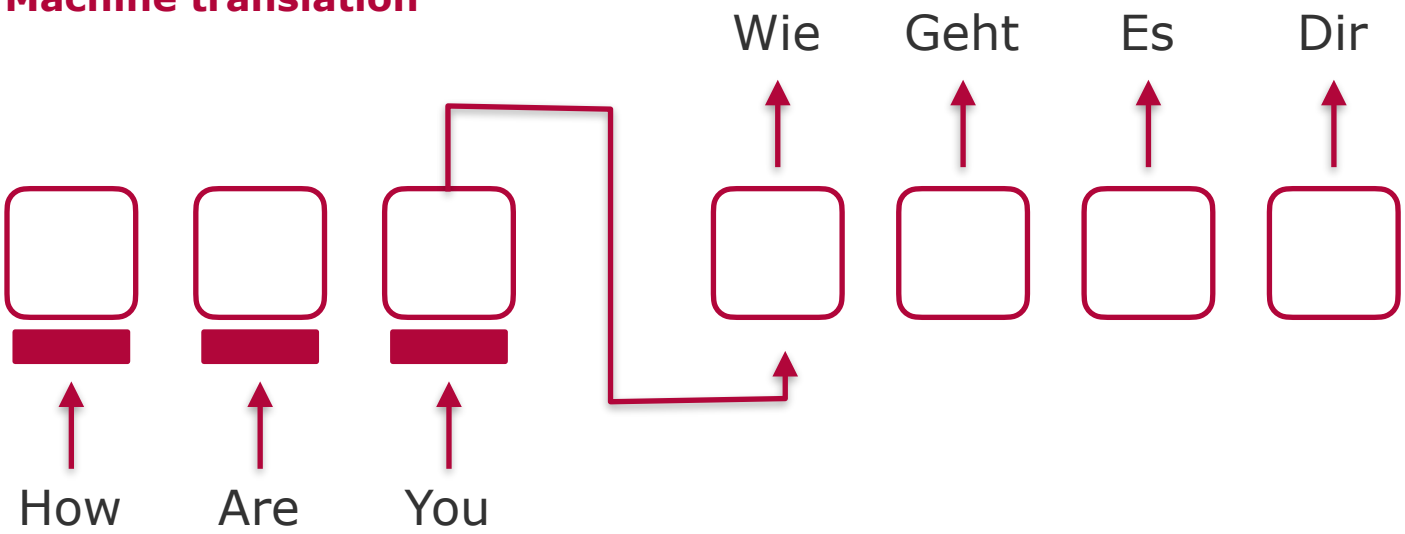
Seq2Seq learning



- Converting a sequence from one domain to another domain
- Reading and writing the tokens one by one

Approach

Machine translation



- Words are embedded first
- Then they are fed into the Seq2Seq model

Approach

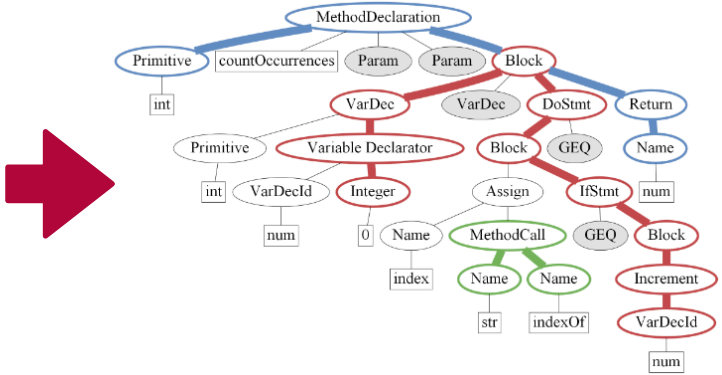
Apply techniques code

- AST structure
- Vector embeddings of code
- Code2Seq learning
- Example: Function name prediction

Approach

AST structure

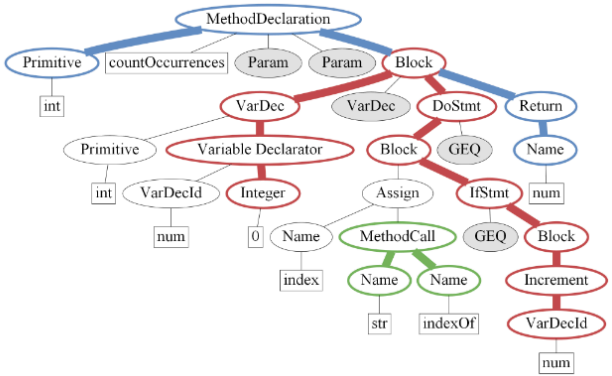
```
int countOccurrences(String str, char ch) {
    int num = 0;
    int index = -1;
    do {
        index = str.indexOf(ch, index + 1);
        if (index >= 0) {
            num++;
        }
    } while (index >= 0);
    return num;
}
```

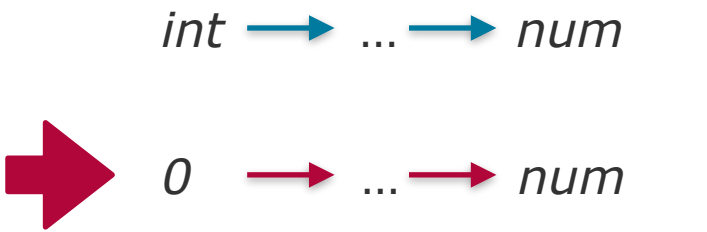


- **Assumption:** AST information is a more reliable predictor for similarity than raw source code

Approach

AST structure

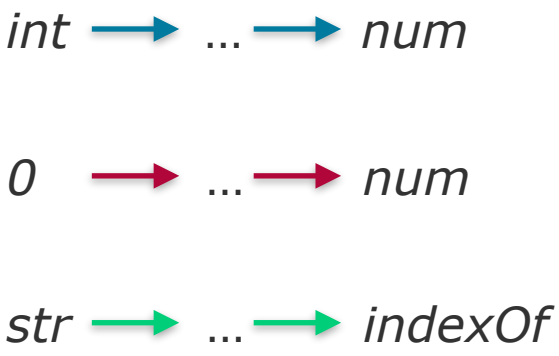




- **Idea:** Sample a set of *k* paths from AST to represent the function

Approach

AST structure



- **Analogy:** Path = Word, Set of *k* paths = Sentence

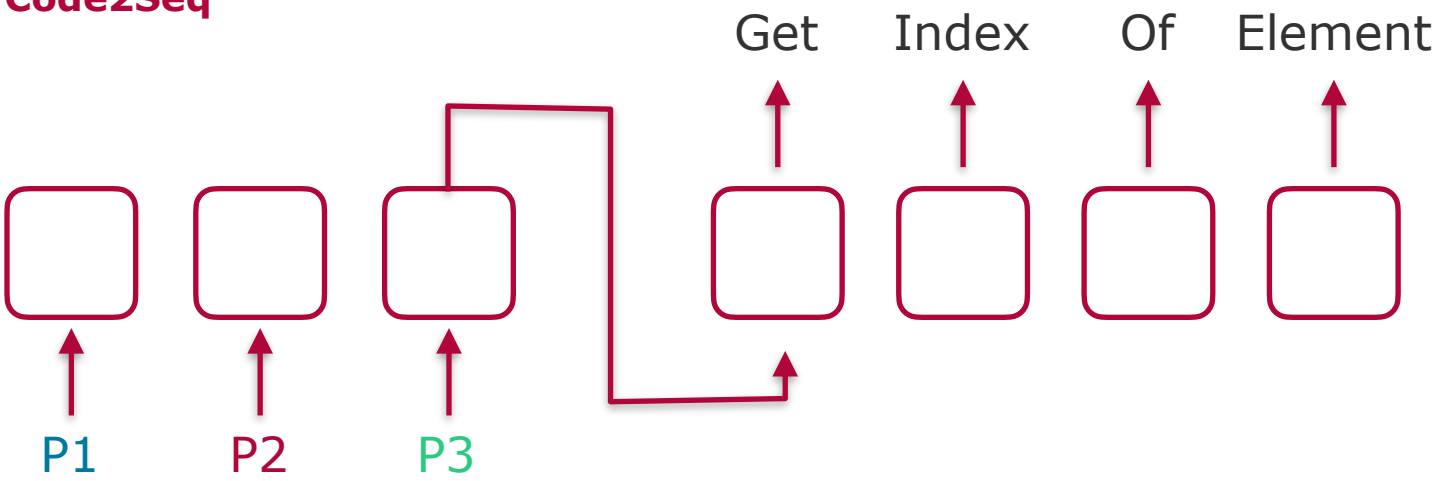
Approach

Vector embeddings of paths

- Each path is mapped into the embedding space
- Dimensions encode abstract semantics

Approach

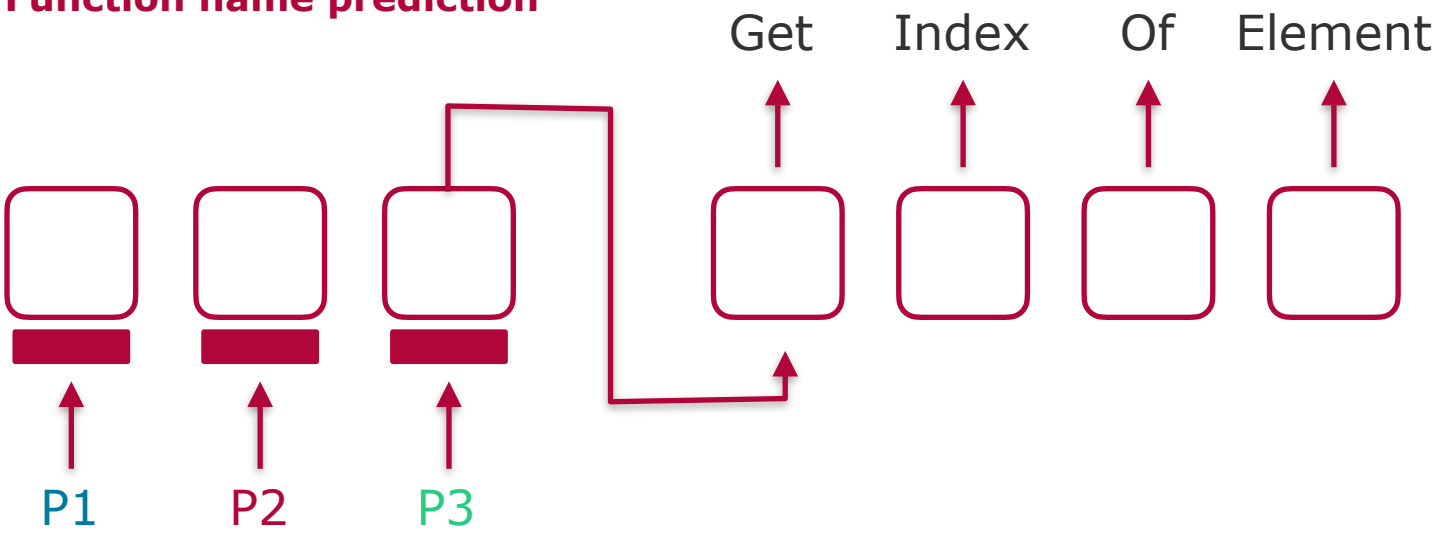
Code2Seq



- Reading a set of paths sampled from the AST
- Predicting the function name token by token

Approach

Function name prediction



- Combination of vector embedding of code and Seq2Seq learning
- More detailed architecture can be found in Appendix A

Approach

Py150 dataset

- Part of the Machine Learning for Programming project
- Consists of parsed ASTs of Python programs from GitHub
- Training: 100,000 files (8GB), Testing: 50,000 files (4GB)
- Encoded in JSON

Approach

Tech Stack

- Python v3.8
- Tensorflow v2.2
- Docker v2.3

Results

Evaluation

- No qualitative metric that indicates how well the predicted name fits the function body
- So far only manual evaluation

Examples

```
def func(elements, value):
    index = -1
    for i, x in enumerate(elements):
        if x == value:
            index = i
    return index
```

- Real name: index_of
- Predicted name: get_element

Results

Examples

```
def func(elements, value):
    count = 0
    for x in elements:
        if x == value:
            count += 1
    return count
```

- Real name: count_occurences
- Predicted name: list_element

Results

Examples

```
def func(elements, value):
    for x in elements:
        if x == value:
            return True
    return False
```

- Real name: contains
- Predicted name: is_element_present

Next Steps

Until the submission

- Better evaluation
- Hyperparameter tuning
- Analysis of embedding space and attention weights
- Train on full data-set

Next Steps

Future work

- Beam search implementation
- Pre-training of embeddings

Conclusion

Solved?

- Proof of concept, tackling only small part of the problems related with stale code
- Promising approach to support documentation
- Can be applied in a range of similar applications: E.g. Code summarisation, documentation, completion, retrieval

Literature

List of image references

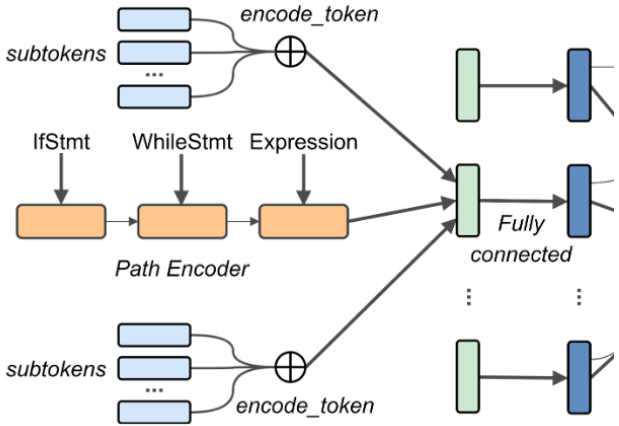
- Word embeddings (https://cdn-images-1.medium.com/max/1600/1*jpnKO5X0Ii8PVdQYFO2z1Q.png)
- Code & AST example (https://arxiv.org/pdf/1808.01400.pdf)
- Transformer architecture (https://www.tensorflow.org/images/tutorials/transformer/transformer.png)

Thank you for your attention!

Lando Löper
Software Architecture Group
21.07.2020

Appendix A

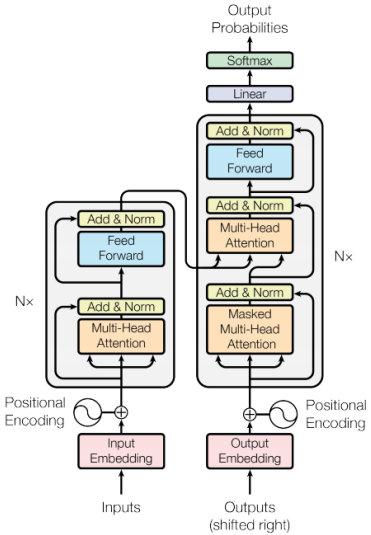
Path Embedding



- Leaves nodes are encoded by the sum of their subtokens
- Path is encoded using a bidirectional RNN
- Finally both are concatenated and fed through a feed forward layer

Appendix A

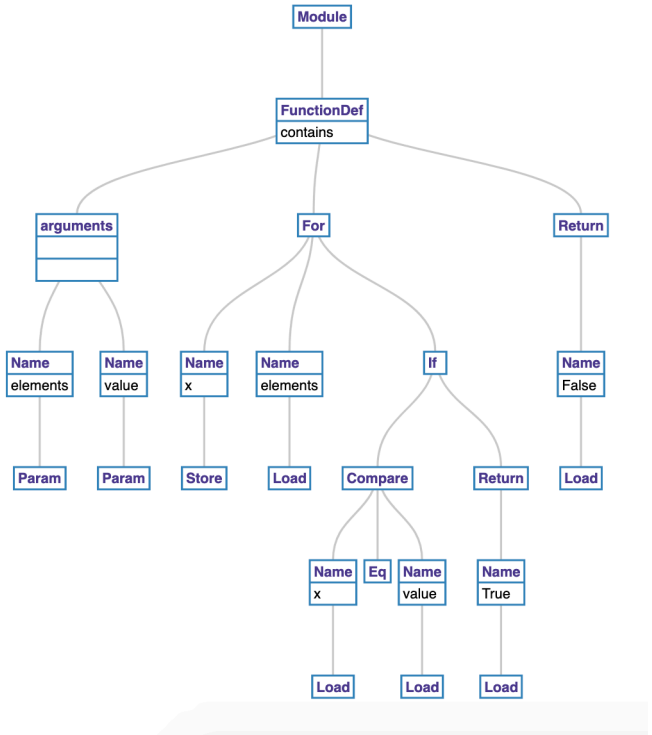
Encoder Decoder Architecture



- The encoder / decoder architecture follows the transformer model
- The path encoder does not apply positional encoding on the set of paths, since they have got no inherent ordering

Appendix B

AST (contains)



Appendix B

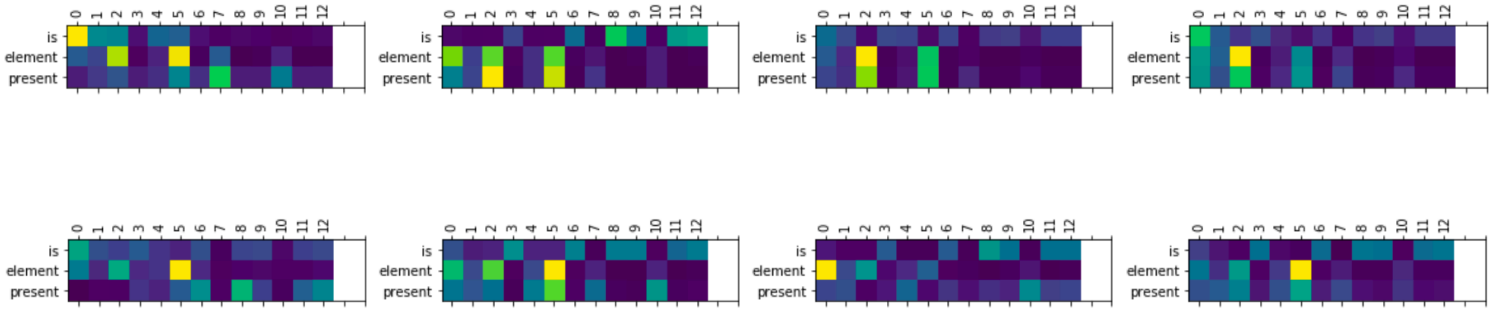
Paths (contains)

```
elements Param|args|Param value
elements Param|args|arguments|FunctionDef|body|For|Store x
elements Param|args|arguments|FunctionDef|body|For|Load elements
elements Param|args|arguments|FunctionDef|body|Return|Load false
value Param|args|arguments|FunctionDef|body|For|Store x
value Param|args|arguments|FunctionDef|body|For|Load elements
value Param|args|arguments|FunctionDef|body|Return|Load false
x Store|For|Load elements
x Store|For|body|Return|Load false
elements Load|For|body|Return|Load false
x Load|CompareEq|Load value
x Load|CompareEq|If|body|Return|Load true
value Load|CompareEq|If|body|Return|Load true
```

- A set of 13 sampled path from the AST
- Structure: leafnode path|between|nodes leafnode

Appendix B

Attention weights (contains)



- Attention weights of the transformer model
- Extracted from both encoder layers and each attention head
- Signalling which of paths was paid most attention to