Topical

Automatic Repository Tagging using Attention on Hybrid Code Embeddings

- Agathe Lherondelle, Varun Babbar, Yash Satsangi, Fran Silavong, Shalteil Eloul, Sean Moran

Machine Learning on Source Code

- Github has ~200 million repositories

- There is a need to search for and categorize repositories in an automated manner.
  - Track popular topics in a large set of repositories
  - Cluster repositories by topics
  - Compute similarities between repositories?
  - Track evolution of a single project

Can we generate repository embeddings for downstream tasks?

Previous Approaches

- Repo2Vec (Rokon et al. 2021):
  - Smaller training dataset used
  - Binary Classification task: Malware vs Non-Malware

- Import2Vec (Theeten, Vandeputte, and Van Cutsem. 2019):
  - Only represents dependencies within script
  - Binary Classification task: Malware vs Non-Malware

What about other code components?

- GraphCodeBERT (Guo et al. 2021):
  - State of the art code representation

But repositories are more than just code!

Topical: Phase 1

For script 1:*N* in repository

Step 1: Generate embeddings for code

Step 2: Generate embeddings for metadata

- Method name
- Script name
- Comments
- Docstrings

Step 3: Generate embeddings for dependency graph

Step 4: Aggregate them!

test_pycg.py

```
from collections import Counter
import time
import logging


def all_unique(nums):
    """returns True if all elements of a list
    are unique otherwise, return False"""
    begin = time.time()
    count = dict(Counter(nums))
    for i in count.values():
        if i > 1:
            logging.info(time.time() - begin)
            return False
    logging.info(time.time() - begin)
    return True
```

Code Embedding

Metadata Embeddings

Code Embedding

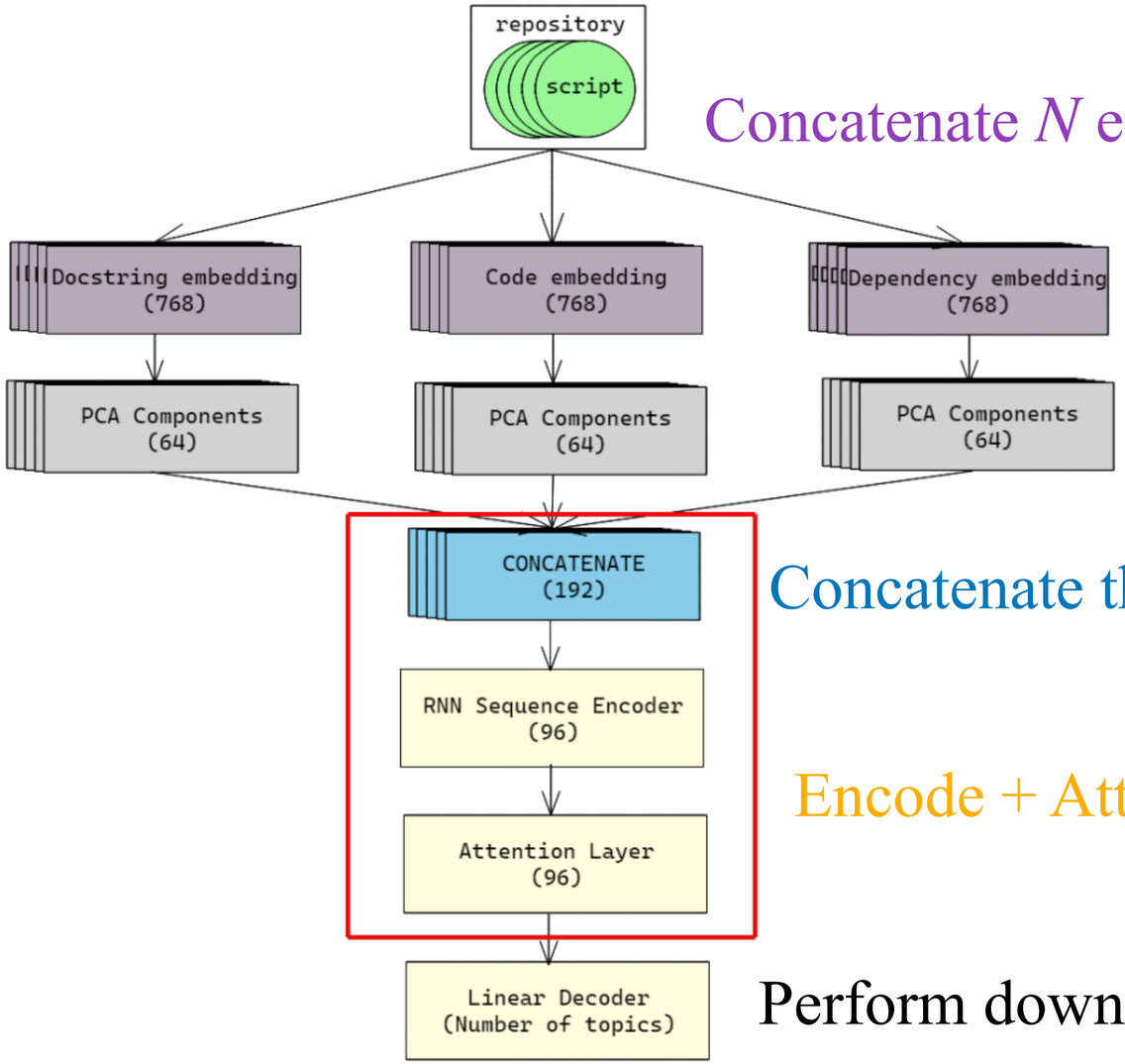
```
{
    "test_pycg": [],
    "test_pycg.all_unique": [
        "collections.Counter",
        "<builtin>.dict",
        "time.time",
        "logging.info"
        ],
    "time.time": [],
    "collections.Counter": [],
    "<builtin>.dict": [],
    "logging.info": []
}
```

Dependency Graph Embedding

- All functions called within the script
  - Built-in functions
  - Functions called in other scripts

Let’s zoom into step 4

How do we aggregate embeddings?



Concatenate *N* embeddings of each type

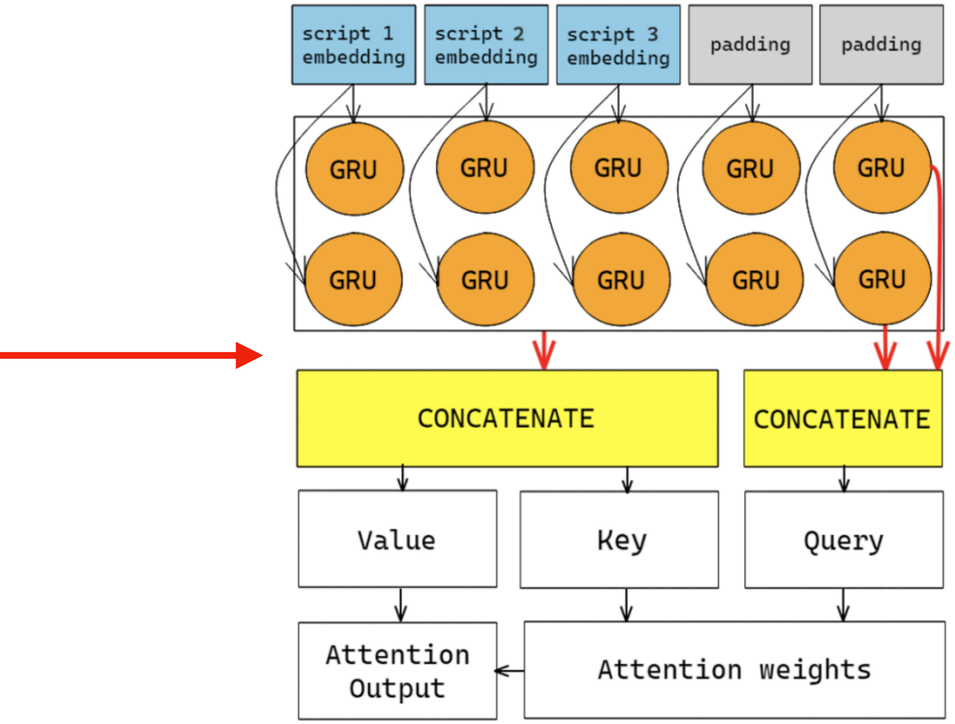
Perform dimensionality reduction for each type

Concatenate the 3 types of embeddings

Encode + Attention

Perform downstream task

Encode + Attention



Can use sequential encoders (GRU / LSTM) if scripts have special ordering

- __init__.py goes first
- main.py last

Alternatively, use MLP-based encoder

We found sequential encoders do better

Apply attention mechanism to encoder output

How do we generate embeddings?

Step 1: Generate embeddings for code

GraphCodeBERT

Step 2: Generate embeddings for metadata

- Method name
- Script name
- Comments
- Docstrings

DistilBERT

Step 3: Generate embeddings for dependency graph

Convert the graph into a list of edges

- Each edge links a script method to its set of dependencies

Embed this list using DistilBERT with special tokens to demarcate edges

*but we can use any kind of embedding
in the Topical framework (e.g. LLM based embeddings)

How does Topical compare to baselines?

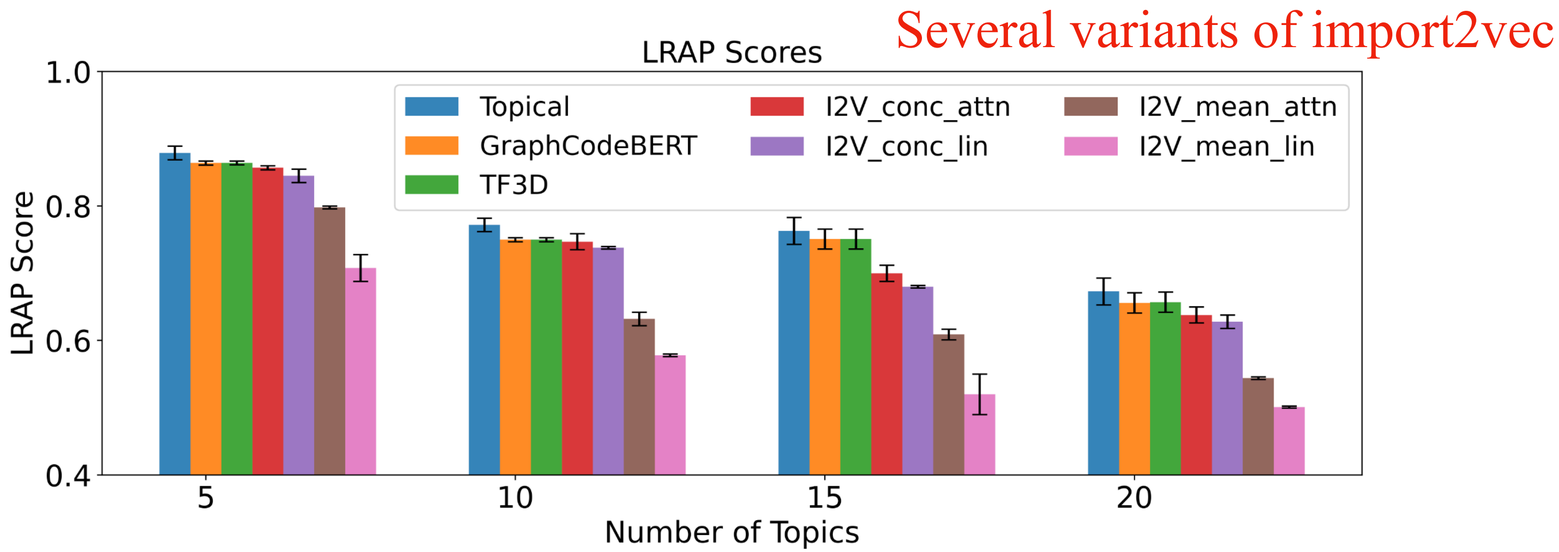
Task: Determine the set of topics associated with a repository

multi-label classification
given a list of 20 topics

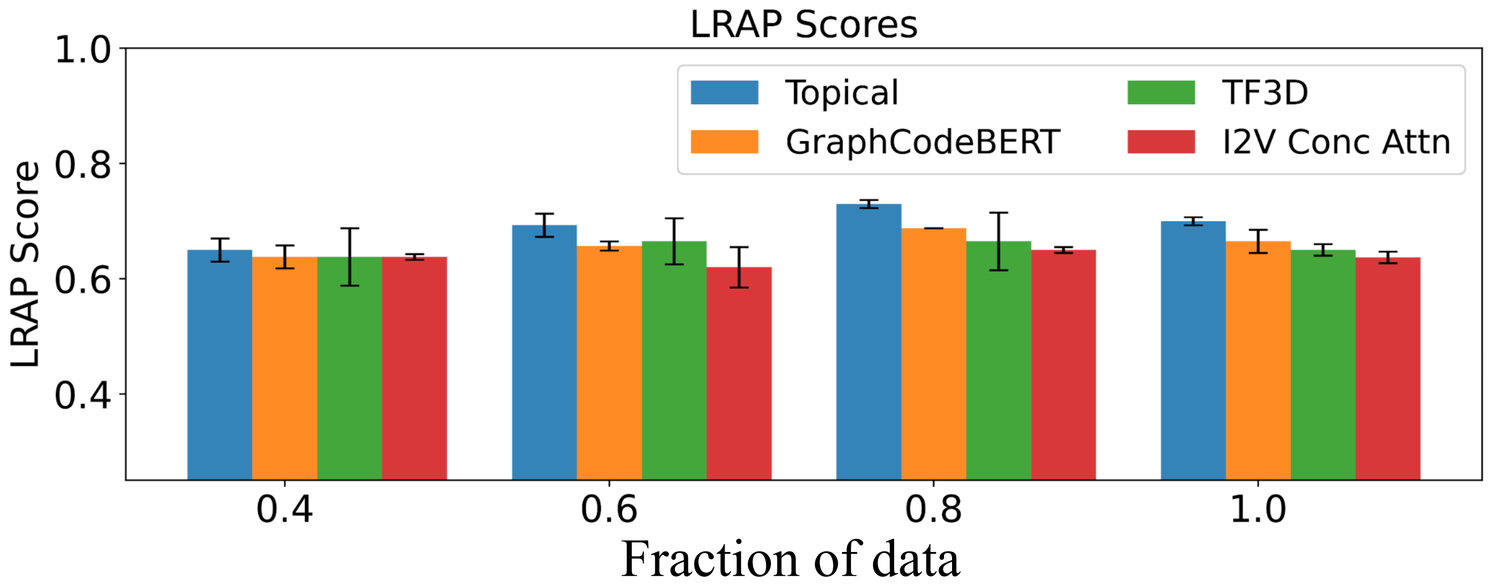
Model	Precision	Recall
Topical	**0.485 ± 0.017**	0.630 ± 0.032
GraphCodeBERT	0.410 ± 0.031	**0.670 ± 0.010**
Import2Vec+Attn	0.350 ± 0.034	0.632 ± 0.034

We beat several competitive methods 

How does Topical compare to baselines?



How does Topical compare to baselines?



Are Topical embeddings coherent? Yes!

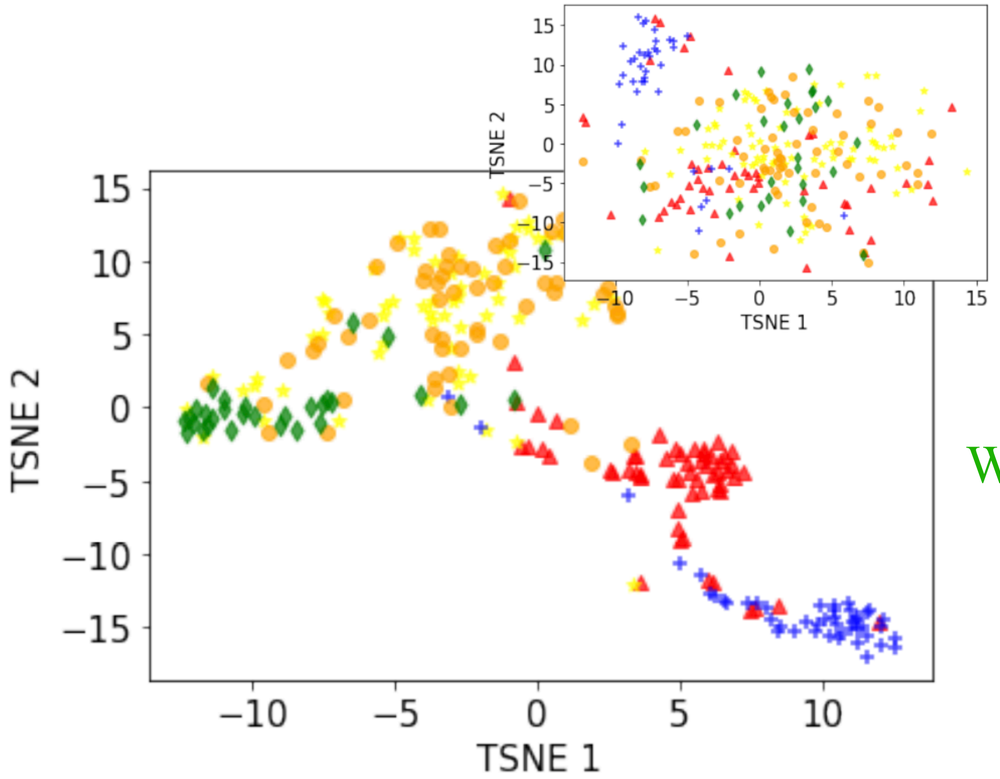
deep learning

machine learning

reinforcement learning

django

databases



Without attention - doesn’t make sense

With attention - coherent clusters seen

Future work

- Tracking the evolution of a project from start to finish
- IR - nearest neighbor search for relevant repositories
- Extend Topical for other programming languages we only used Python
- Benchmark IR and topic classification performance with LLM based embeddings

Thank you!