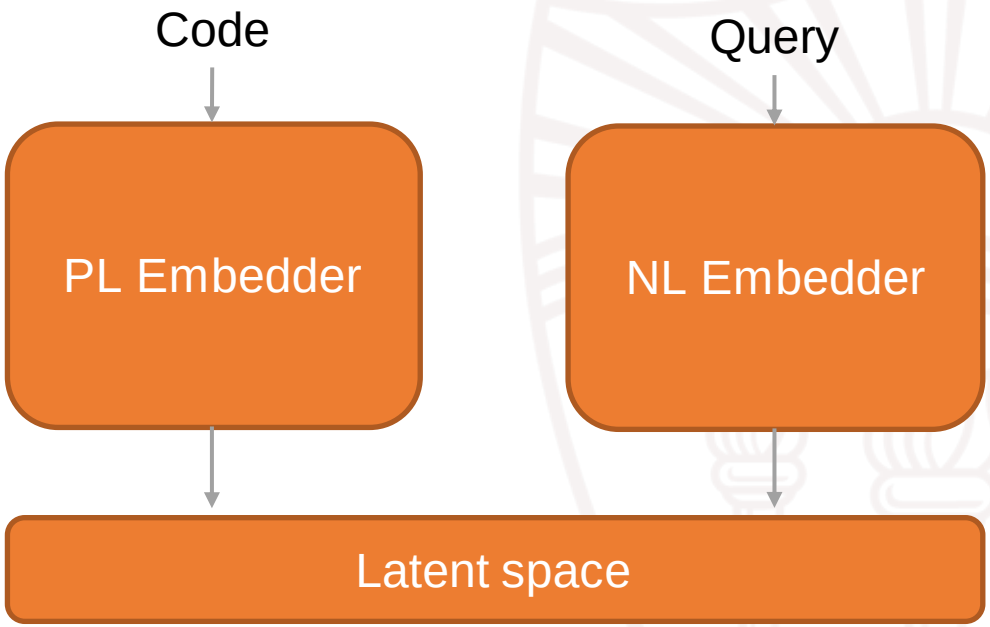
USC University of Southern California

NS3: Neuro-Symbolic Semantic Code Search

Presented by Shushan Arakelyan

USC

Semantic Code Search

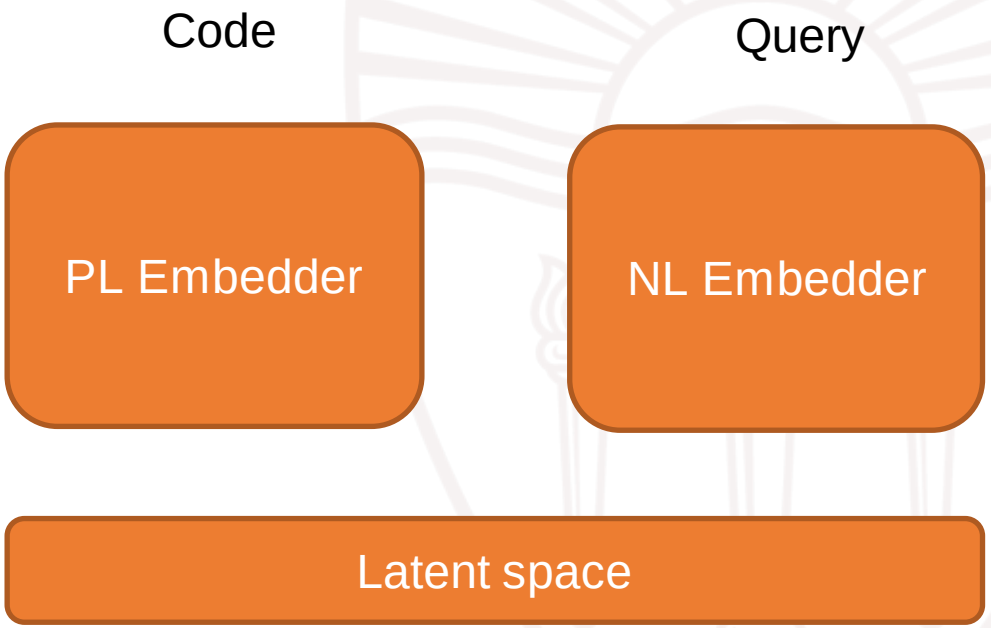


Limitations

1. Faithfully capturing details in the queries
2. Inability to perform multi-step reasoning

Semantic Code Search

Pretraining large language models that have proven to work well on textual tasks

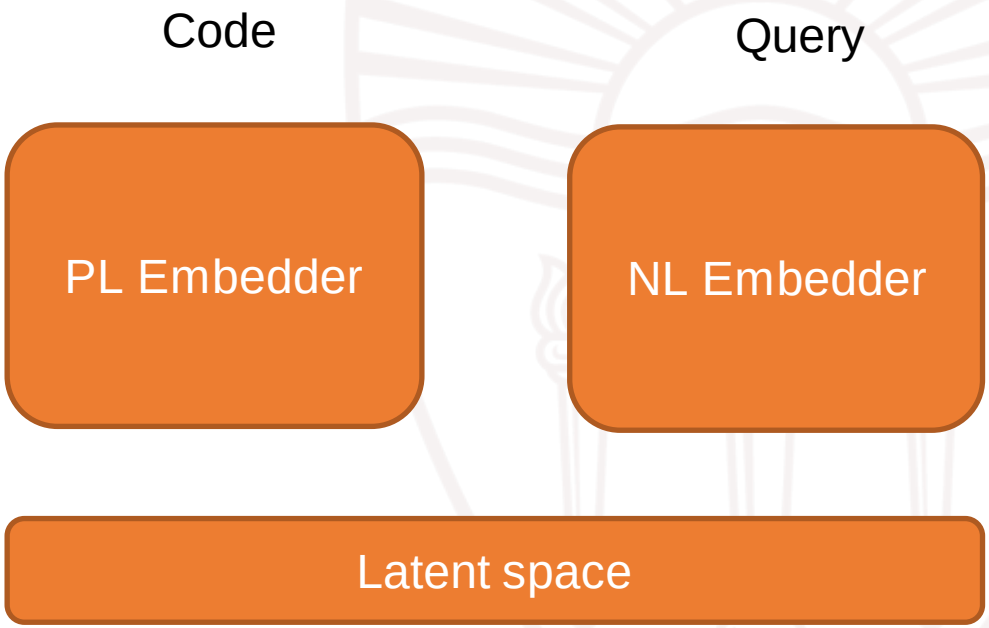


BERT/GPT → CuBERT, CodeBERT and CodeGPT

USC

Semantic Code Search

Enriching the code embedder to take into consideration unique properties of code



CodeBERT → GraphCodeBERT

USC

Motivation

Task A: 1) *Are references to lists present?*

Task A: 1) ✓
Task B: 1) ✓2) ✓3) ✓

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):                    ← 1
        for j in range(0, n-i-1):           ← 1
            if arr[j] > arr[j + 1] :        ← 2
                arr[j], arr[j + 1] =        ← 3
                    arr[j + 1], arr[j]
```

Task A: 1) ✓
Task B: 1) ✓2) ✓3) ×

```
def findMax(arr):
    n = len(arr)
    m = arr[0]
    for i in range(n-1):                    ← 1
        if arr[i] > m:                      ← 2
            m = arr[i]
```

USC

Motivation

Task A: 1) *Are references to lists present?*

Task B: 1) *Are iterations over lists present?*

Task A: 1) ✓

Task B: 1) ✓2) ✓3) ✓

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):                    ← 1
        for j in range(0, n-i-1):           ← 1
            if arr[j] > arr[j + 1] :        ← 2
                arr[j], arr[j + 1] =        ← 3
                    arr[j + 1], arr[j]
```

Task A: 1) ✓

Task B: 1) ✓2) ✓3) ×

```
def findMax(arr):
    n = len(arr)
    m = arr[0]
    for i in range(n-1):                    ← 1
        if arr[i] > m:                      ← 2
            m = arr[i]
```

USC

Limitations of current approaches

1. Capturing long or compositional queries

2. Inability to perform multi-step reasoning

Motivation

Task A: 1) *Are references to lists present?*

Task B: 1) *Are iterations over lists present?*

Task B: 2) *Are there comparisons of elements?*

Task A: 1) ✓

Task B: 1) ✓ 2) ✓ 3) ✓

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):                    ← 1
        for j in range(0, n-i-1):           ← 1
            if arr[j] > arr[j + 1] :        ← 2
                arr[j], arr[j + 1] =
                    arr[j + 1], arr[j]      ← 3
```

Task A: 1) ✓

Task B: 1) ✓ 2) ✓ 3) ×

```
def findMax(arr):
    n = len(arr)
    m = arr[0]
    for i in range(n-1):                    ← 1
        if arr[i] > m:                      ← 2
            m = arr[i]
```

USC

Motivation

Task A: 1) *Are there references to array or list?*

Task B: 1) *Are iterations over lists present?*

2) *Are there comparisons of elements?*

3) *Are elements swapped to produce sorted result?*

Task A: 1) ✓

Task B: 1) ✓2) ✓3) ✓

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):                    ← 1
        for j in range(0, n-i-1):           ← 1
            if arr[j] > arr[j + 1] :        ← 2
                arr[j], arr[j + 1] =
                    arr[j + 1], arr[j]      ← 3
```

Task A: 1) ✓

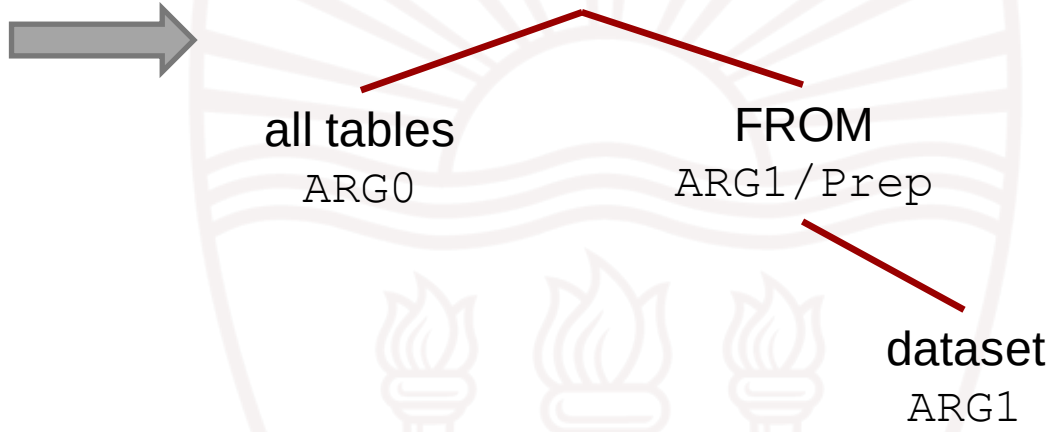
Task B: 1) ✓2) ✓3) ×

```
def findMax(arr):
    n = len(arr)
    m = arr[0]
    for i in range(n-1):                    ← 1
        if arr[i] > m:                      ← 2
            m = arr[i]
```

USC

Semantic Structure of the Query

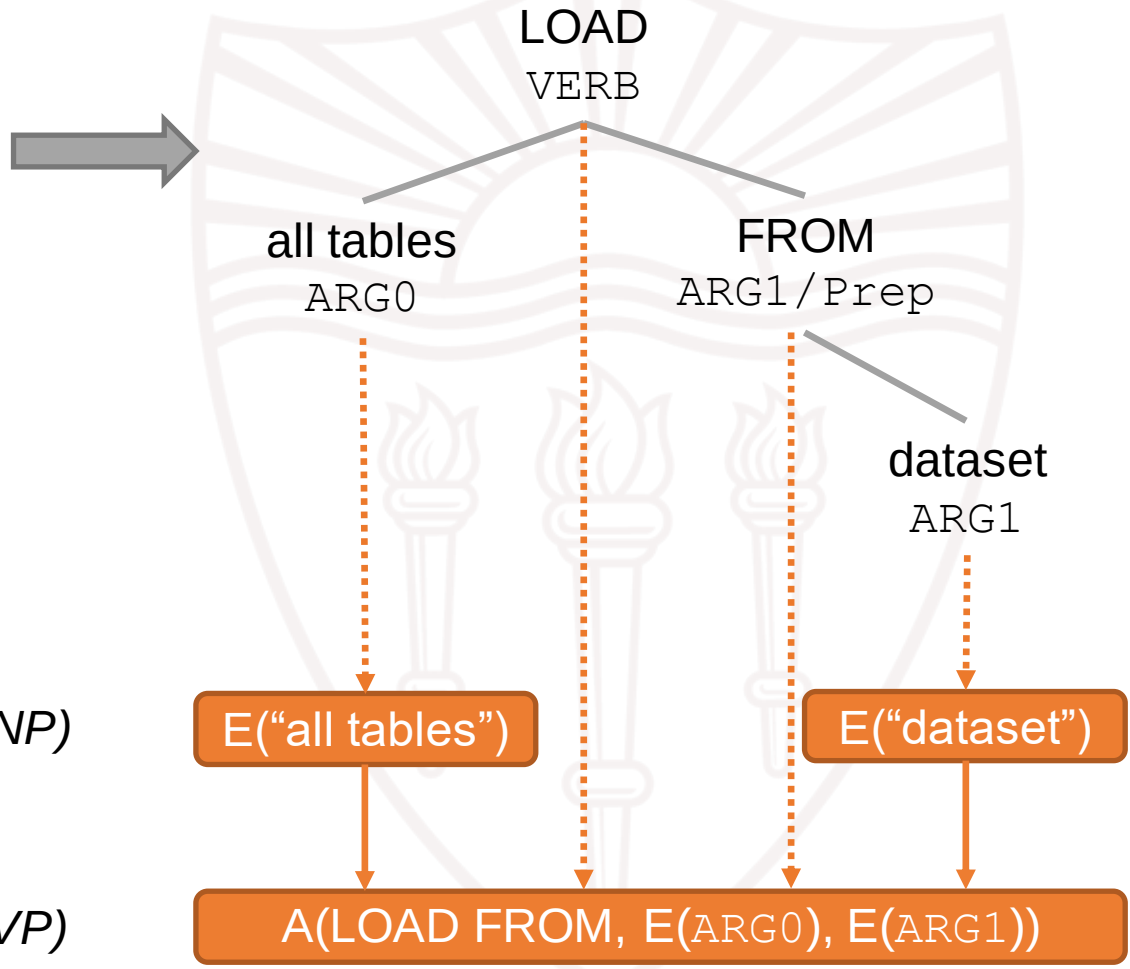
"Load all tables from dataset"



USC

Semantic Structure of the Query

"Load all tables from dataset"



E for discovering entities *(N/NP)*

A for discovering actions *(V/VP)*

USC

Modules

Entity discovery:
"dataset"

Action:
"Load ??? from dataset"

IN

Entity "***dataset***"

Tokenized code $[c_1, \ldots c_N]$

Joint representation for verb and preposition: ***"load from"***

Scores for code tokens: $[e_1, e_2, \ldots, e_N]$

Tokenized code $[c_1, \ldots c_N]$

OUT

Scores for code tokens: $[e_1, e_2, \ldots, e_N]$

Estimate for output of E(***"all tables"***)

USC

Results, Pt. 1

Method	CSN				CSN-10K				CSN-5K			
	MRR	P@1	P@3	P@5	MRR	P@1	P@3	P@5	MRR	P@1	P@3	P@5
BM25	0.209	0.144	0.230	0.273	0.209	0.144	0.230	0.273	0.209	0.144	0.230	0.273
RoBERTa (code)	0.842	0.768	0.905	0.933	0.461	0.296	0.545	0.664	0.29	0.146	0.324	0.438
CuBERT	0.225	0.168	0.253	0.294	0.144	0.081	0.166	0.214	0.081	0.03	0.078	0.118
CodeBERT	0.873	0.803	0.939	0.958	0.69	0.550	0.799	0.873	0.680	0.535	0.794	0.870
GraphCodeBERT	0.812	0.725	0.880	0.919	0.786	0.684	0.859	0.901	0.773	0.677	0.852	0.892
NS^3	**0.924**	**0.884**	**0.959**	**0.969**	**0.826**	**0.753**	**0.886**	**0.908**	**0.823**	**0.751**	**0.881**	**0.913**

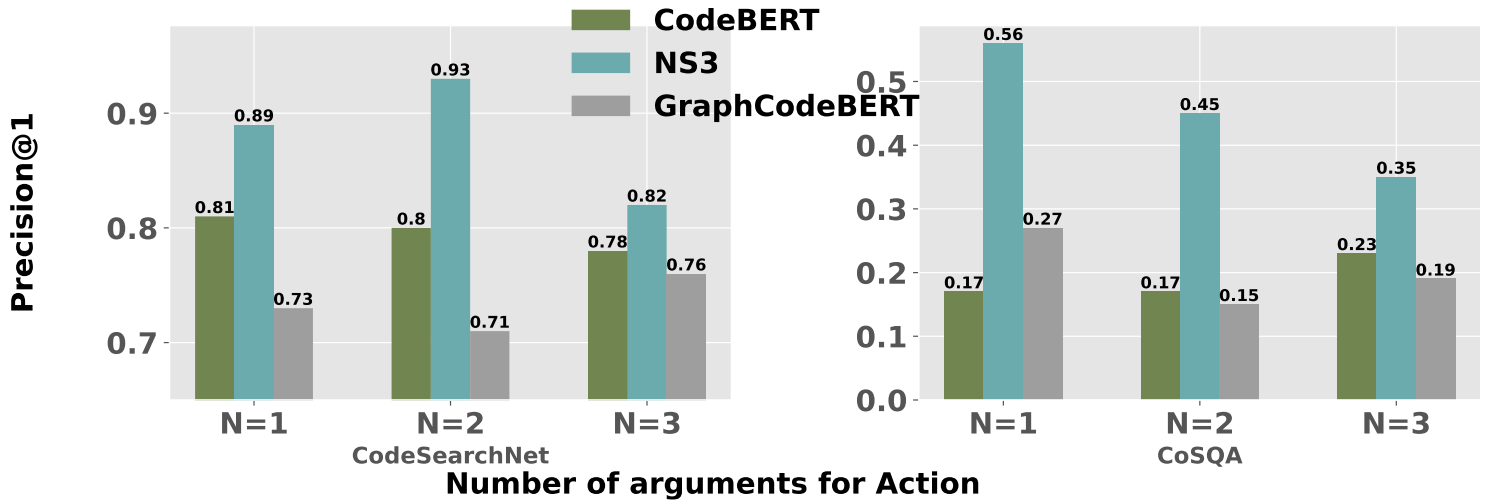
Method	CoSQA			
	MRR	P@1	P@3	P@5
BM25	0.103	0.05	0.119	0.142
RoBERTa (code)	0.279	0.159	0.343	0.434
CuBERT	0.127	0.067	0.136	0.187
CodeBERT	0.345	0.175	0.42	0.54
GraphCodeBERT	0.435	0.257	0.538	0.628
NS^3	**0.551**	**0.445**	**0.619**	**0.668**

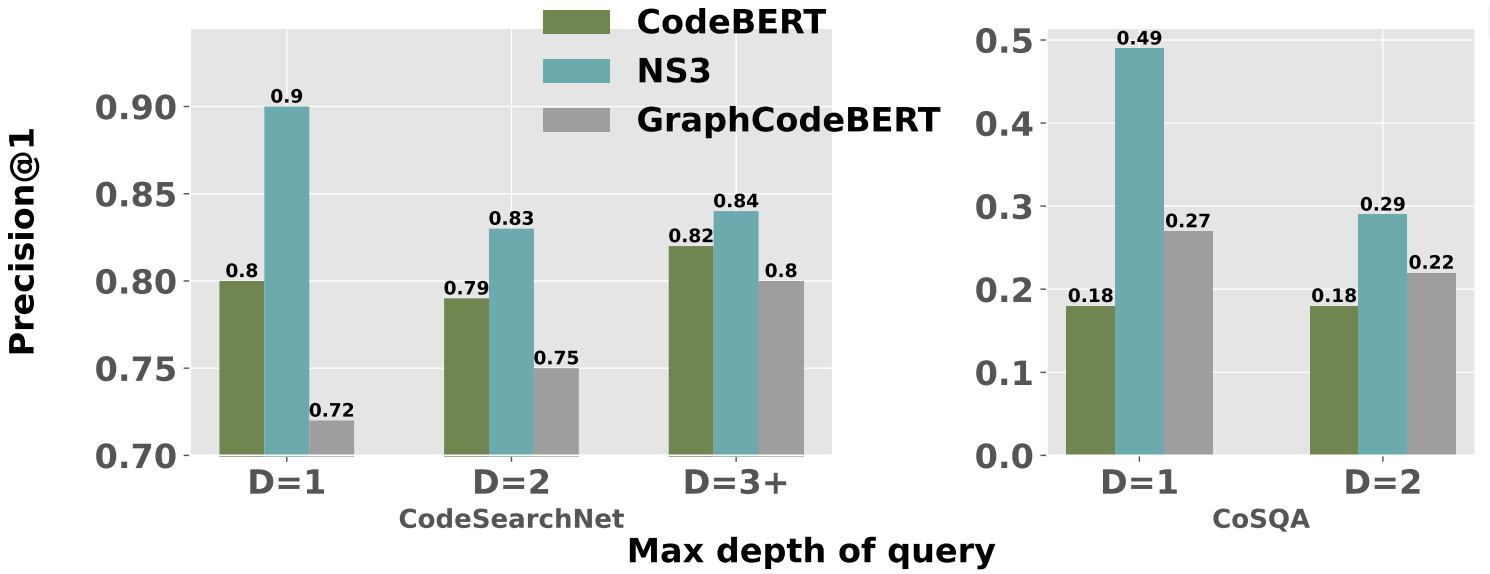
Solid improvement over baselines in all evaluation setups.

USC

Results, Pt. 2

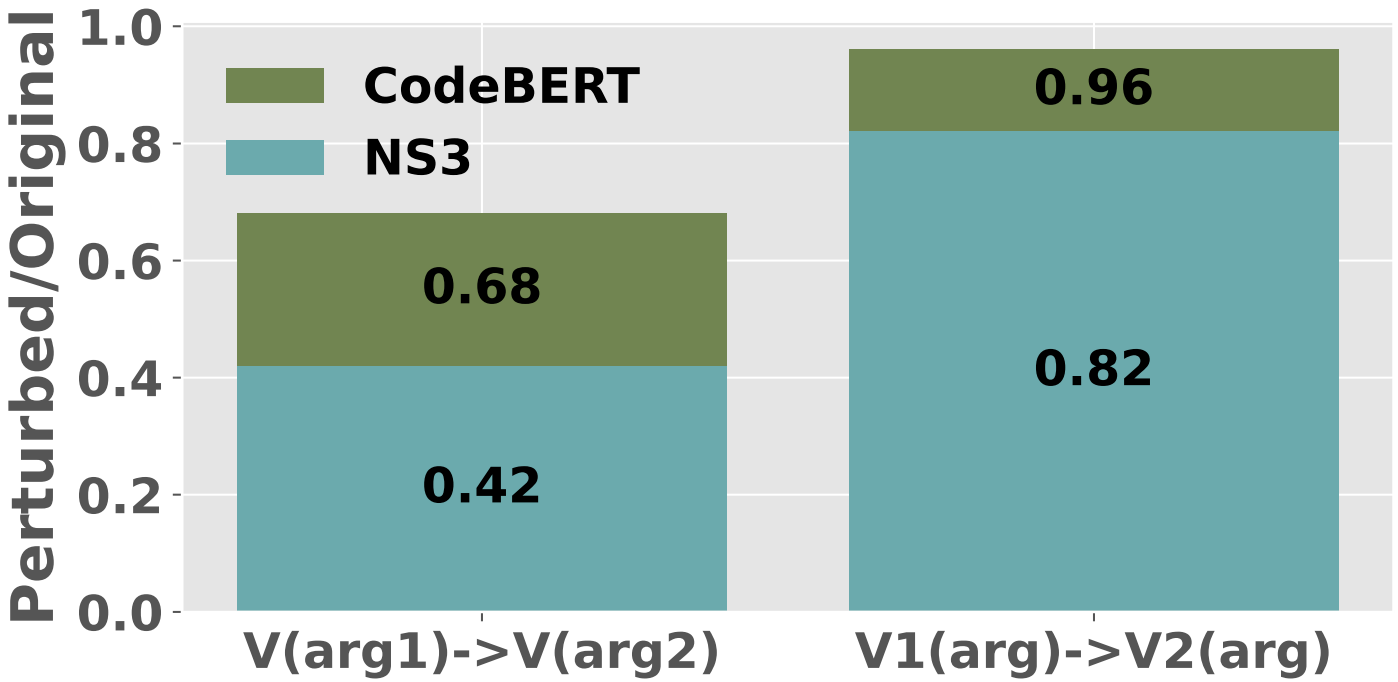
Improved performance for deeper and wider queries





USC

Results, Pt. 3



In a more sensitive model, the similarity score for query-code pair should drop after perturbation of the query, thus leading to lower ratio value.

USC

Results, Pt. 4

End-to-end training leaves us with good semantic relevance scores for the entity discovery module

Entity = "***redundant elements***"

```
def dedup_list(l):
    dedup = set()
    return [x for x in l if not (\
        x in dedup or dedup.add(x))]
```

After pretraining

```
def dedup_list(l):
    dedup = set()
    return [x for x in l if not (\
        x in dedup or dedup.add(x))]
```

After finetuning

Action = ***"Remove ? of list"***

```
def dedup_list(l):
    dedup = set()
    return [x for x in l if not (\
        x in dedup or dedup.add(x))]
```

Entity = "***folders***"

```
def get_all_files(folder):
    for path, dirlist, filelist \
     in os.walk(folder):
        for fn in filelist:
            yield op.join(path, fn)
```

After pretraining

```
def get_all_files(folder):
    for path, dirlist, filelist \
     in os.walk(folder):
        for fn in filelist:
            yield op.join(path, fn)
```

After finetuning

Action = ***"Navigate ?"***

```
def get_all_files(folder):
    for path, dirlist, filelist \
     in os.walk(folder):
        for fn in filelist:
            yield op.join(path, fn)
```

Thank you!

Anna Hakhverdyan

Miltiadis Allamanis

Luis Garcia

Christophe Hauser

Xiang Ren

USC

USC University of Southern California

NEURAL INFORMATION PROCESSING SYSTEMS

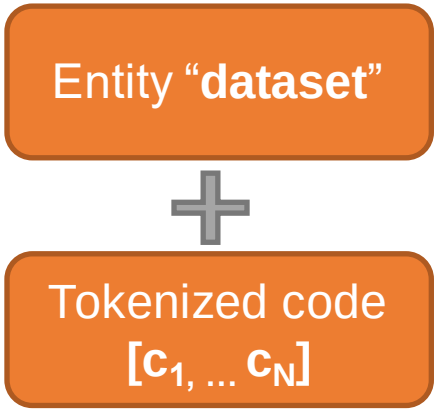
NS^3: Neuro-Symbolic Semantic Code Search

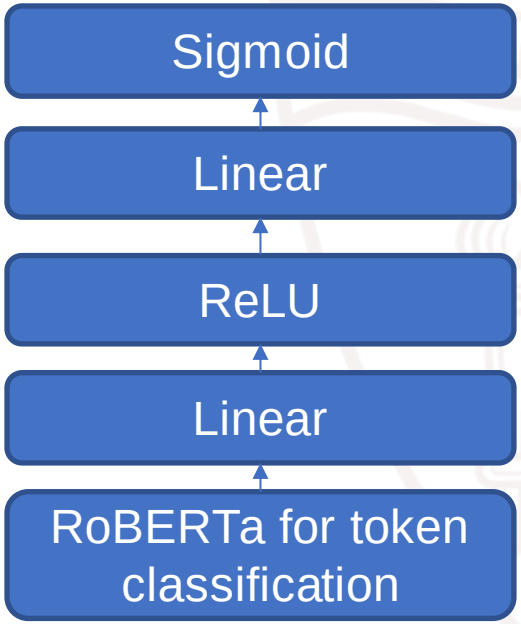





USC

Modules





$[e_1, e_2, ..., e_N]$

Scores for code tokens

USC

USC

USC

USC