Multi-task Deep Learning in the Software Development domain

Silvia Severini, Garching, 04.11.19
Advisor: Ahmed Elnaggar and Davide Bacciu

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline

- **Introduction**
- Background
- Research questions
- Approach
- Datasets
- Results and discussion
- Conclusion
- Future research

The Software Development Domain

High-quality software:

- nowadays almost every aspect of life relies on it (health, transportation, entertainment)
- Costly
- Hard-work for developers

=> Application of **Machine Learning** techniques

“Deep learning has achieved competitive performance against previous algorithms on about 40 SE tasks” [cit]

Problem

When applying Deep learning to software development domain tasks, we face three main problems:

- Data scarcity
- Energy consumption
- Manipulation of source code

Source code Python:

```python
from pygithub3 import Github
username = raw_input("Please enter a Github username: ")
password = raw_input("Please enter the account password: ")
gh = Github(login=username, password = password)
get_user = gh.users.get()
user_repos = gh.repos.list().all()
for repo in user_repos:
    print repo.language
```

Description:
Getting repository information using pygithub3 for Python

Source code summarization for Python

Outline

- Introduction
- **Background**
- Research questions
- Approach
- Datasets
- Results and discussion
- Conclusion
- Future research

Natural Language Processing on Source Code

- Context awareness
- Unlimited vocabulary
- Data preprocessing and tokenization

```java
public class HelloWorld {

    public static void main(String[] args) {
        // Prints "Hello, World" to the terminal window.
        System.out.println("Hello, World");
    }

}
```

public class HelloWorld { public static void main (String [ ] args) { System . out . println (" Hello, World ") ; } }

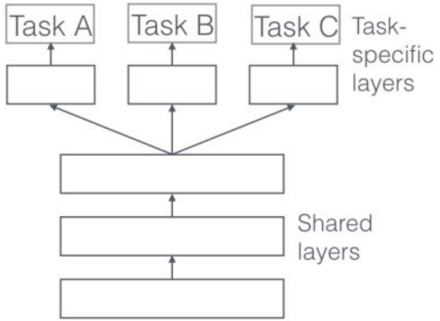
Multi-task Deep Learning

(Multi-task Deep Learning) *given m learning tasks $\{T_i\}_{i=1}^m$ where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for T_i by using the knowledge contained in all or some of the m tasks.*

Why?

- Implicit data augmentation
- Attention focusing
- Representation bias
- Regularization

=> Augment of the generalization capabilities



Hard parameter sharing for multi-task learning in deep neural networks [1]

Outline

- Introduction
- Background
- **Research questions**
- Approach
- Datasets
- Results and discussion
- Conclusion
- Future research

Research questions

1. **Can multi-task deep learning be beneficial for tasks in the software development domain?**
2. **How far is multi-task deep learning from state-of-the-art solutions in the software development domain?**
3. **Could the model be trained with the English language and programming languages together?**
4. **How does training on multiple tasks of the software development domain simultaneously compare to training on each task separately act?**

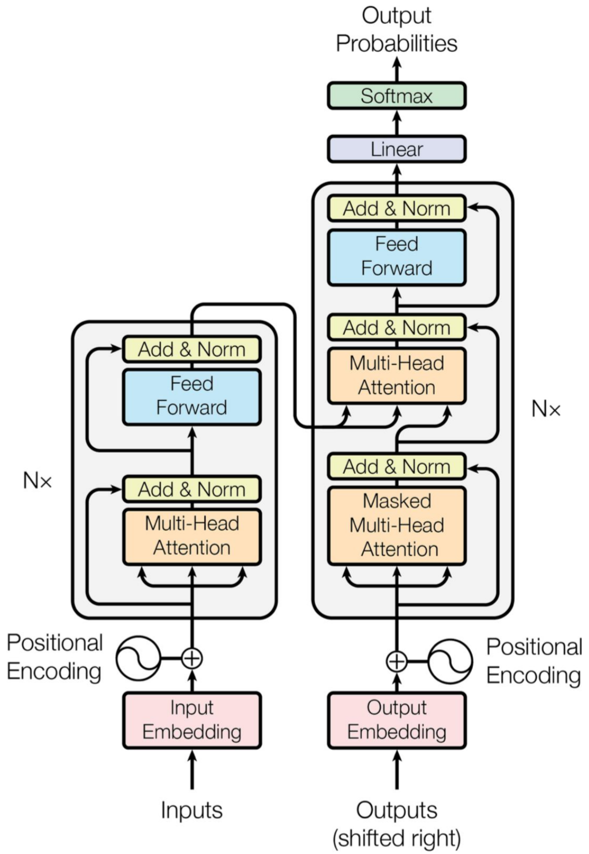
Outline

- Introduction
- Background
- Research questions
- **Approach**
- Datasets
- Results and discussion
- Conclusion
- Future research

Transformer model

State-of-the-art for sequence-to-sequence problem manipulation:

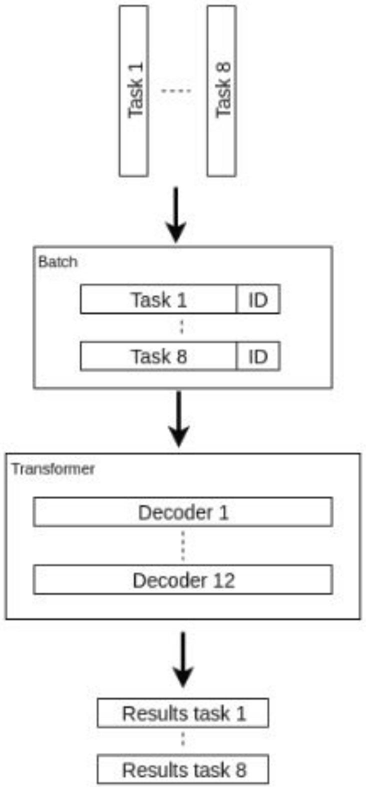
- Parallelizable computation
- Faster training
- Manipulation of long-range dependencies
- Encoder - Decoder



Transformer model by Vaswani et al. (2017)

Model architecture

- Multi-task
  - Decoder-only with attention
  - Vocabulary from the language model
  - Multiple loss functions optimized concurrently
- Single-task
  - Each task is trained independently
  - No language model objective
  - Encoder-decoder with attention



Experimental setup

Hardware:

Machine name	P100	DGX-1
GPUs	1x NVIDIA P100	8x Tesla V100
GPU Core	3.5 K	41 K
Memory	16 GB	8 x 16 GB

Leibniz-Rechenzentrum (LRZ) available machines

Tensor2Tensor

- TensorFlow based library maintained by the Google Brain team

- Three phases:
  - Data generation
  - Training
  - Decoding and evaluation

Outline

- Introduction
- Background
- Research questions
- Approach
- **Datasets**
- Results and discussion
- Conclusion
- Future research

Dataset

#	Tasks in requirement (1 paper)
A1	Requirement extraction from natural languages (1)

#	Tasks in design (1 papers)
B1	Design pattern recognition (1)

#	Tasks in development (30 papers)
C1	**Program learning and program synthesis (14)**
C2	Automatic software repair (2)
C3	Code suggestion (2)
C4	Knowledge unit linking in Stack Overflow (2)
C5	**Autonomous driving software (1)**
C6	API description selection (1)
C7	**API sequence recommendation (1)**
C8	Cross-lingual question retrieval (1)
C9	Code comment generation (1)
C10	Commit message generation (1)
C11	**Hot path prediction (1)**
C12	Just-in-time defection prediction (1)
C13	**Model visualization (1)**
C14	Source code summarization (1)

#	Tasks in Testing (27 papers)
D1	Defect prediction (9)
D2	**Reliability or changeability estimation (8)**
D3	Deep learning testing (3)
D4	Energy consumption estimation (1)
D5	**Grammar-based fuzzing testing (1)**
D6	Retesting necessity estimation (1)
D7	Reliability model selection (1)
D8	Robot testing (1)
D9	**Test input generation for mobile (1)**
D10	Testing effort estimation (1)

#	Tasks in maintenance (27 papers)
E1	**Malware detection (10)**
E2	Bug localization (4)
E3	Clone detection (3)
E4	**System anomaly prediction (2)**
E5	Workload prediction in the cloud (2)
E6	Bug report summarization (1)
E7	Bug triager (1)
E8	**Duplicate bug report detection (1)**
E9	**Feature location (1)**
E10	Real-time task scheduling (1)
E11	Test report classification (1)

#	Tasks in management (12 papers)
F1	**Development cost or effort estimation (6)**
F2	Source code classification (4)
F3	Software size estimation (1)
F4	Traceable link generation (1)

Industrial practitioners participate in 13 SE tasks (21 papers)

- C1: *DeepMind, Facebook, Google, Microsoft* (8 papers)
- C5: *Fiat Chrysler Automobiles* (1)
- C7: *Microsoft* (1)
- C11: *Clinc Inc.* (1)
- C13: *Facebook* (1)
- D2: *URU Video, Inc.* (1)
- D5: *Microsoft* (1)
- D9: *IBM* (1)
- E1: *Baidu, Microsoft* (2)
- E4: *Tencent Corporation* (1)
- E8: *Accenture Tech.* (1)
- E9: *ABB Corporate* (1)
- F1: *Motorola Canada Ltd.* (1)

Dataset and preprocessing

1. 7 supervised tasks from *“Deep learning in software engineering” [Li et al.]*
2. Unsupervised Language Model with 5 languages

Task	# samples
Source Code Summarization Python	15.000
Source Code Summarization C#	60.000
Source Code Summarization SQL	29.000
Code comment generation Java	527.400
Commit messages generation	30.000
API sequence recommendation	7.500.000
Program learning and synthesis	90.000

- Substitution of char, string and numbers with specific tokens
- Linearization of the code snippets
- Specific Tokenizer for each language

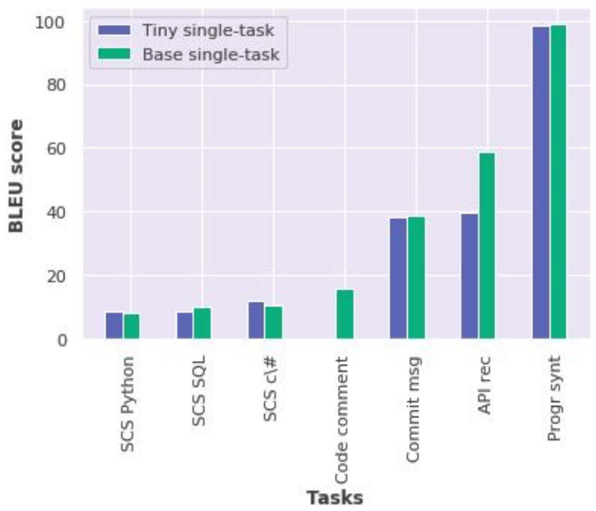
Dataset	# samples
English: 1 Billion world corpus	300.000.000
150K Python Dataset	150.000
SQL corpus	133.000
Java from PGA	700.000
C# from PGA	500.000

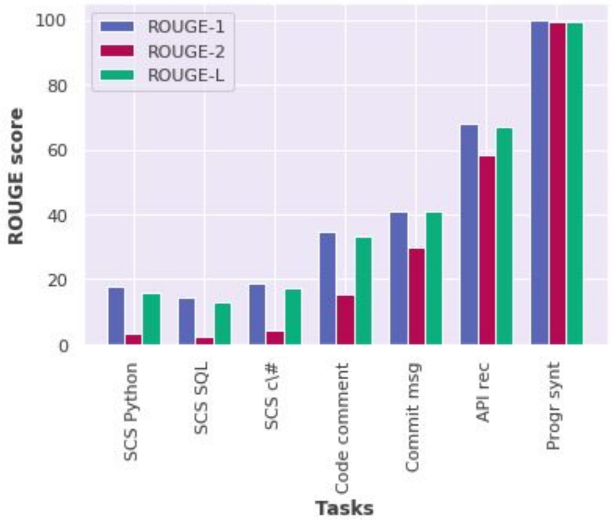
Outline

- Introduction
- Background
- Research questions
- Approach
- Datasets
- **Results and discussion**
- Conclusion
- Future research

Single-task learning

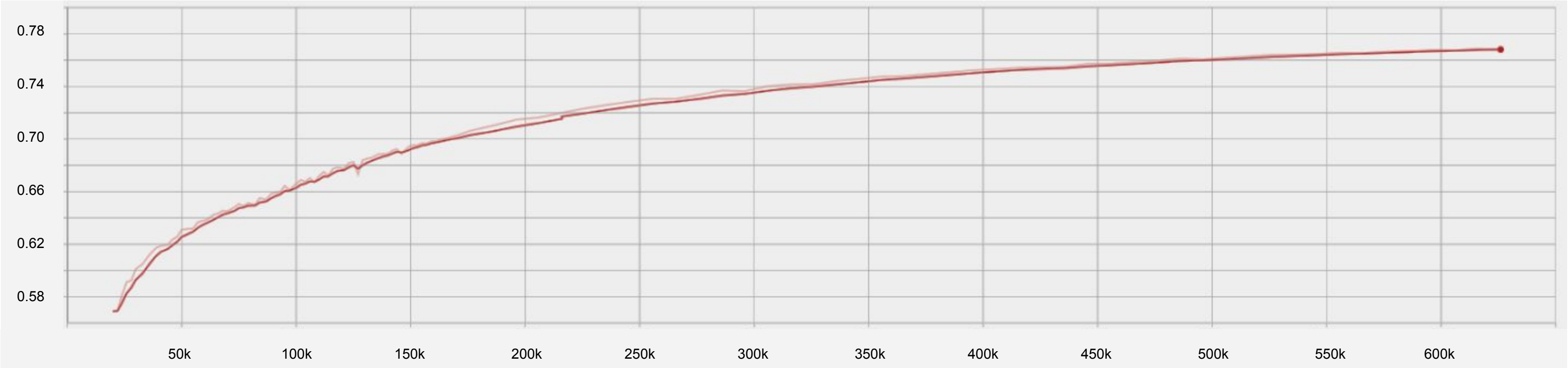
Model	Base	Tiny
Vocabulary size	8192	8192
Hidden size	512	128
Batch size	4096	256
Maximum sequence length	1024	1024
Number of parameters	$\sim$ 50M	$\sim$ 25M
GPUs used	1	1





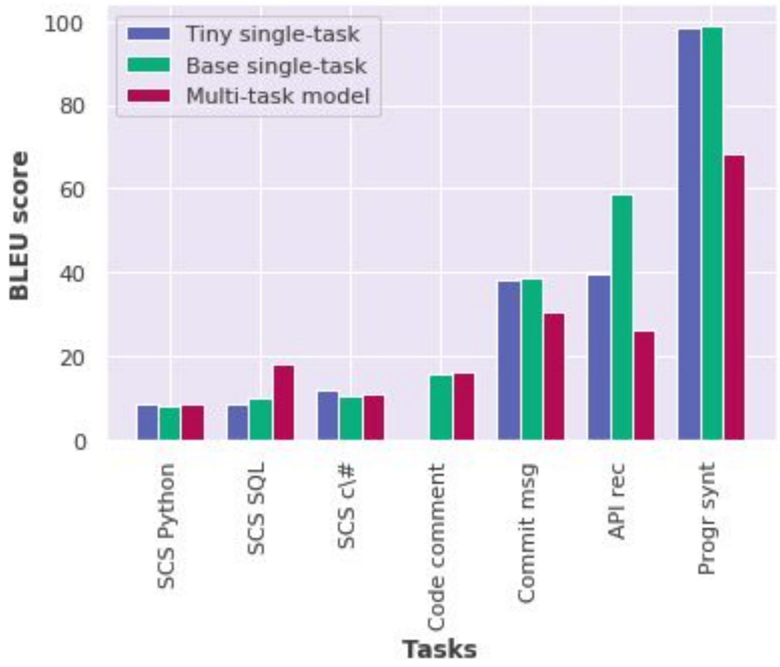
Multi-task learning

Vocabulary size	64K
Hidden size	12
Batch size	1024
Maximum sequence length	1024
Number of parameters	360M
GPUs used	8



Single-task vs Multi-task

- MTL performed better on summarization tasks
- Overfitting avoided for MTL

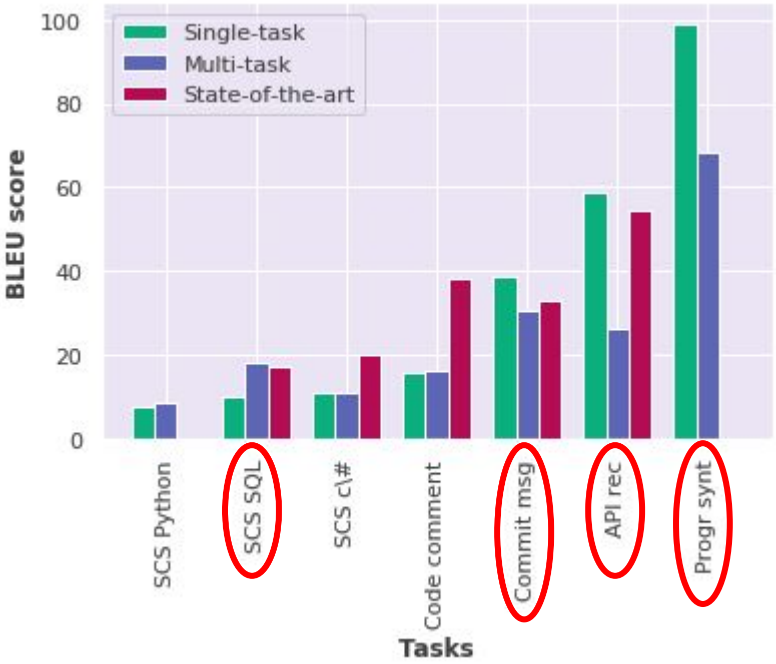


Comparison with state-of-the-art

- First and last do not have BLEU counterpart
- Improvement over 4 tasks

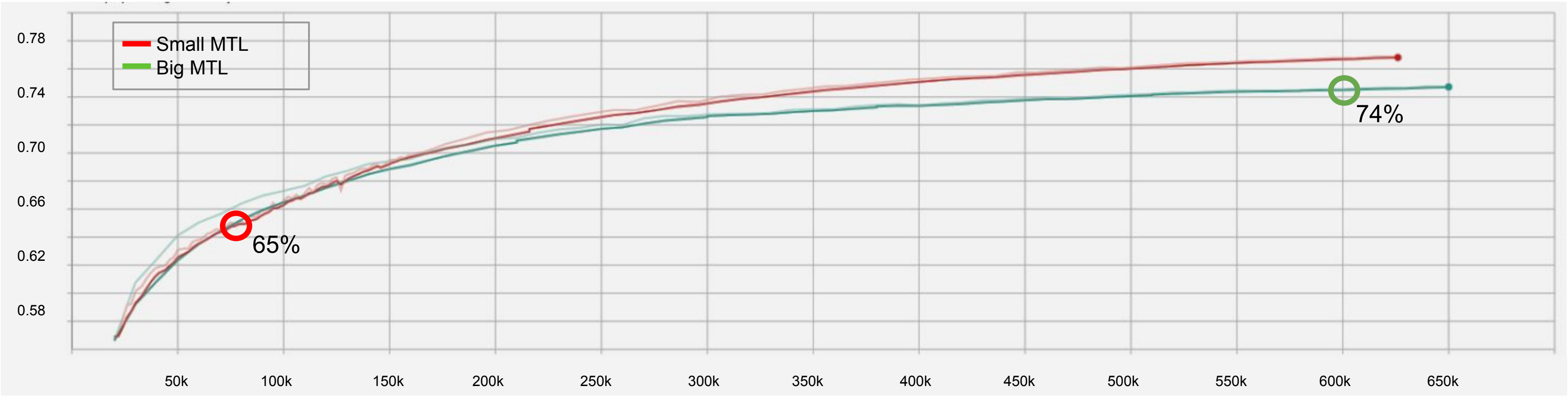
Program Synthesis

Single-task	99.87%
Multi-task	94.8%
State-of-the-art	85.8%



Partial results

Vocabulary size	64K
Hidden size	12
Batch size	128
Maximum sequence length	1024
Number of parameters	422M
GPUs used	8



Outline

- Introduction
- Background
- Research questions
- Approach
- Datasets
- Results and discussion
- **Conclusion**
- Future research

Conclusions

- We investigate the application of Multi-task learning in the software development domain
- We trained single-task and multi-tasks models based on the Transformer architecture
- Consider the structural information of the source code to deal with English and programming languages
- We achieved better performances than the state-of-the-art papers on 4 tasks
- Multi-task learning is promising for this domain, given enough resources available

Outline

- Introduction
- Background
- Research questions
- Approach
- Datasets
- Results and discussion
- Conclusion
- **Future research**

Future research

- Deeper hyperparameters exploration
- Combination of different tasks to understand which give benefits
- Better evaluate the effectiveness of the language model
- Starting point for future research in the software development domain

TUM sebis

Silvia Severini

Technische Universität München
Faculty of Informatics
Chair of Software Engineering for
Business Information Systems

Boltzmannstraße 3
85748 Garching bei München

Tel +49.89.289. 17132
Fax +49.89.289.17136

matthes@in.tum.de
wwwmatthes.in.tum.de