Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

即时缺陷预测技术研究

张钊

2022年12月11日

内容提要

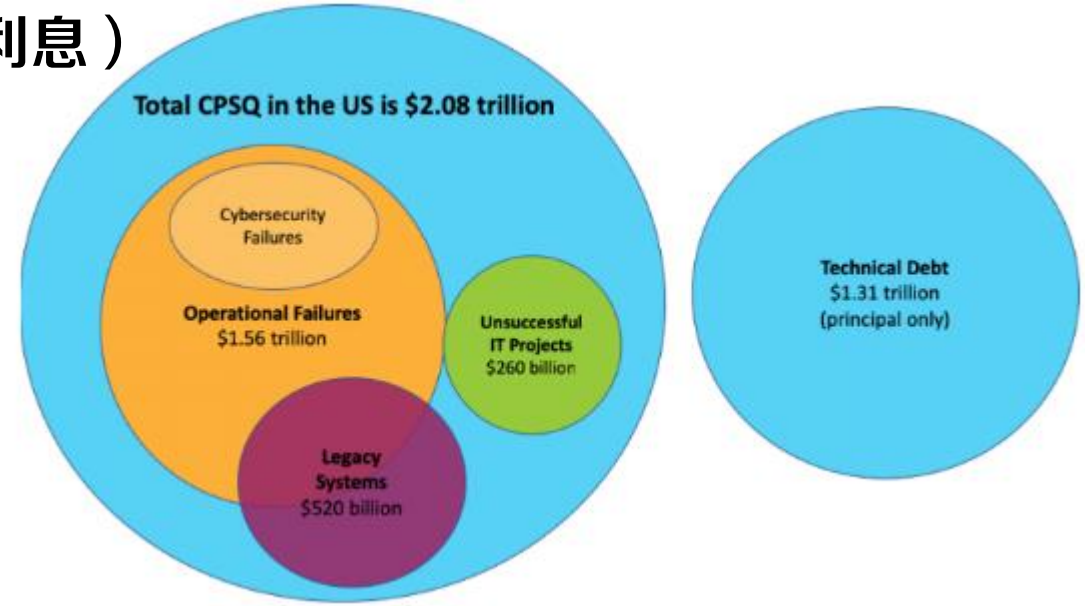
- 背景简介
- 基本概念
- 算法原理
- 总结
- 参考文献

背景简介

- 预期收获
  - 了解即时缺陷预测技术的发展历史
  - 理解即时缺陷预测技术的基本方法
  - 理解缺陷预测、缺陷定位的概念及相关原理

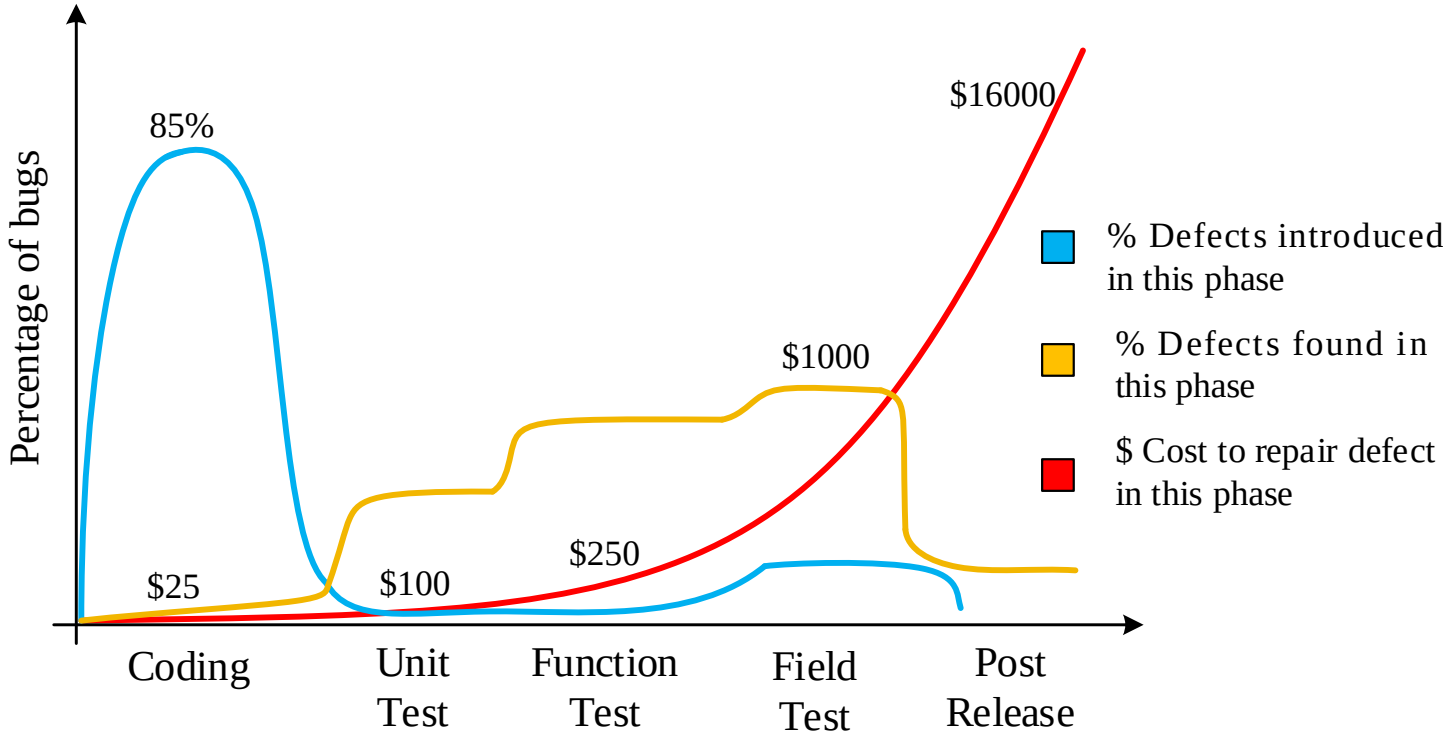
背景简介

- 根据信息和软件质量联盟（CISQ）报告：
  - 2020年开始，大量企业进行数字化转型，软件创新和开发规模进一步扩张
  - 软件数量迅猛增长，但其质量未得到保障。软件质量远远落后于标准，导致软件使用企业额外的负担和财务损失
  - 仅2020 年，美国因软件质量不过关就已付出了2.08 万亿美元的成本代价。如果将2020年美国软件技术债务中存在的严重缺陷修复，还需消耗 1.31 万亿美元（不包括利息）



背景简介

- 缺陷修复
  - 软件维护中的关键活动，但它同时需要消耗大量的时间和资源
  - 数据表明，修复缺陷所需要的成本占软件开发总成本的 50%~75%
- 及时发现缺陷，有助于减少修复缺陷的成本，提高软件质量



背景简介

- 软件缺陷预测
  - 目的：提前预测可能存在缺陷的软件实体（如文件和变更）
  - 基本思路：从软件项目的历史数据中提取特征用来表征被预测的软件实体，然后将这些特征输入到分类（或回归）器进行训练获得预测模型，从而对新产生的软件实体预测其存在缺陷的可能性
  - 研究意义：
    - 优化资源分配，节省维护成本
    - 及时发现缺陷，提高代码质量

背景简介

- 传统的缺陷预测技术针对粗粒度的软件实体，例如文件、模块或包进行预测。
  - 软件缺陷预测模型可能会预测一个巨大的文件存在缺陷，但是对于开发者来说，查看整个文件来寻找缺陷非常耗费精力和时间
  - 这样一个巨大的文件可能被很多开发者修改过，因此，寻找一个合适的开发者来对这个文件检查也是一个非常困难的任务

背景简介

- 即时缺陷预测（ Just-in-Time defect prediction ）
  - 目的：预测开发者每次提交的代码变更（ code change ）是否存在缺陷
  - 基本思路：开发者提交一次代码变更后即对变更代码进行缺陷分析，预测其存在缺陷的可能性
    - 细粒度：变更级预测关注更加细粒度的软件实体
    - 即时性：即时缺陷预测技术可以在代码变更提交时进行预测
    - 易追溯：开发者提交的代码变更中保存了开发者的信息

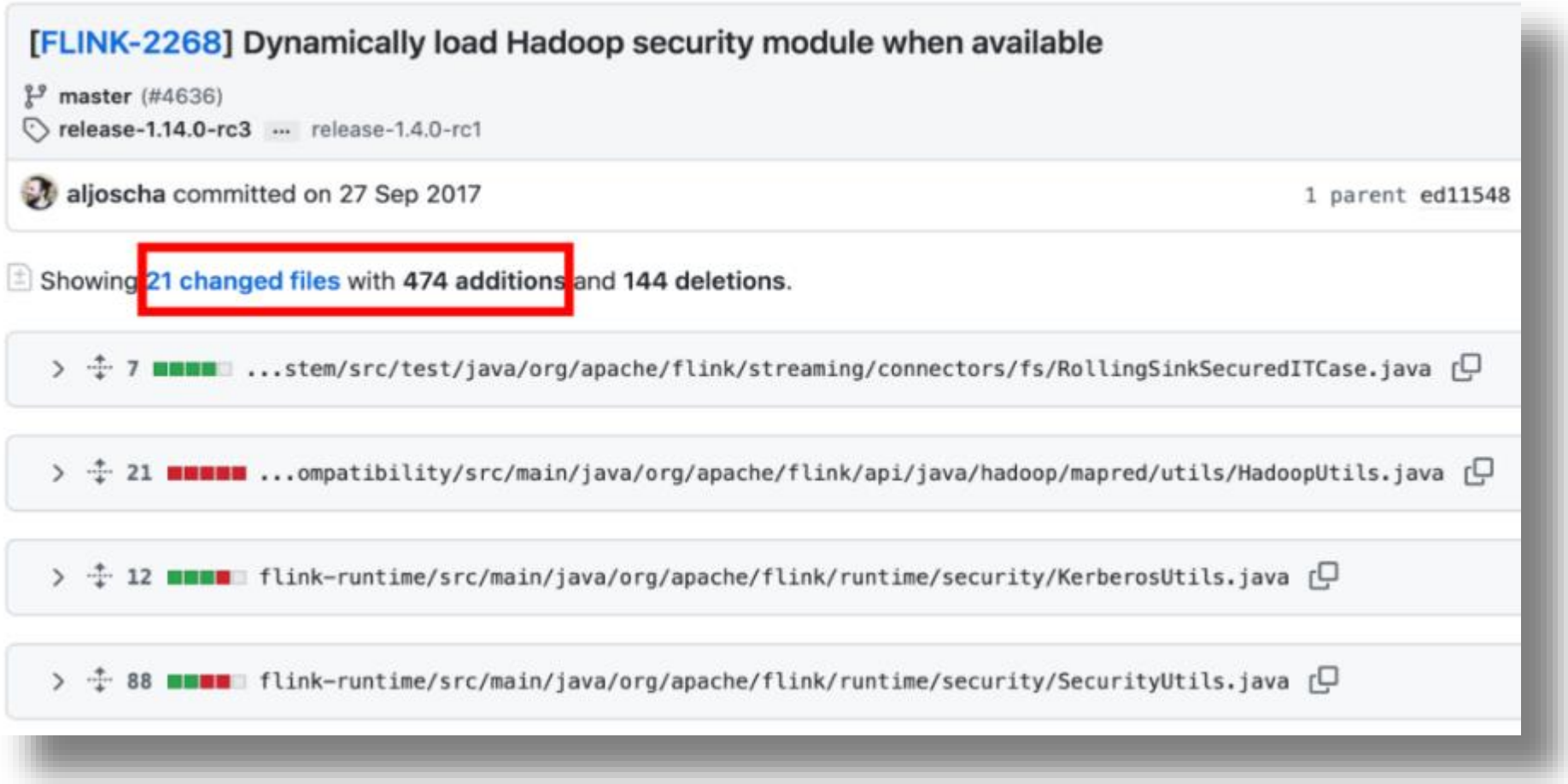
基本概念

基本概念

- 软件缺陷
  - 根据 IEEE 标准定义，软件缺陷是指软件产品中存在的，导致产品无法满足软件需求和其规格要求，需要进行修复的瑕疵、问题。
  - 软件缺陷的存在，极大制约了软件的应用与发展，带来了大量的经济损失

基本概念

- 代码变更
  - 对软件源代码的增加，删除，修改



JIT-Fine

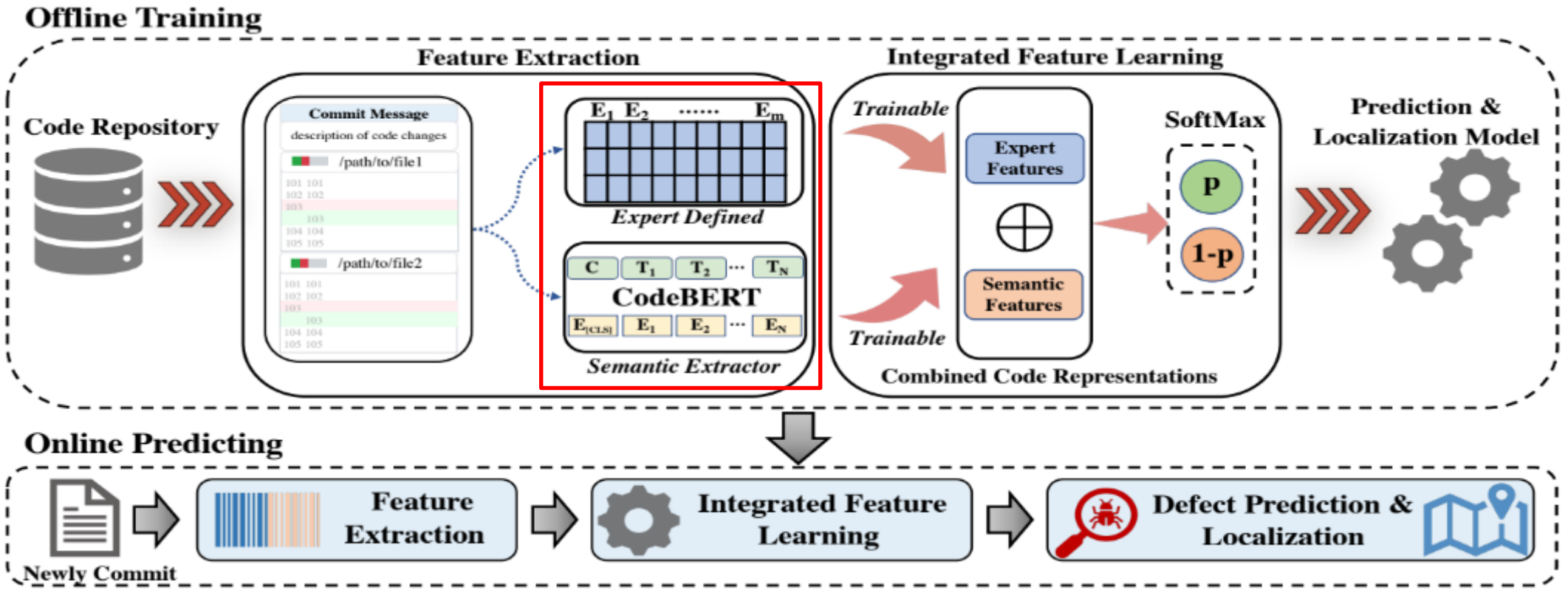
JIT-Fine

T	目标	识别存在缺陷的代码提交，在代码行级别定位缺陷位置
I	输入	提交的软件代码变更（提交信息、添加的代码行、删除的代码行）
P	处理	1. 特征提取，特征包括利用CodeBERT生成的嵌入向量和14个专家特征 2. 集成特征学习，利用全连接网络合并训练语义特征和专家特征 3. 缺陷预测和定位
O	输出	软件代码变更是否存在缺陷及缺陷引入代码行

P	问题	软件缺陷预测的粗粒度及低准确率
C	条件	缺陷引入提交中标记行标签
D	难点	如何实现高准确率的软件缺陷预测和代码行级缺陷定位
L	水平	ESEC/FSE2022（CCF-A）

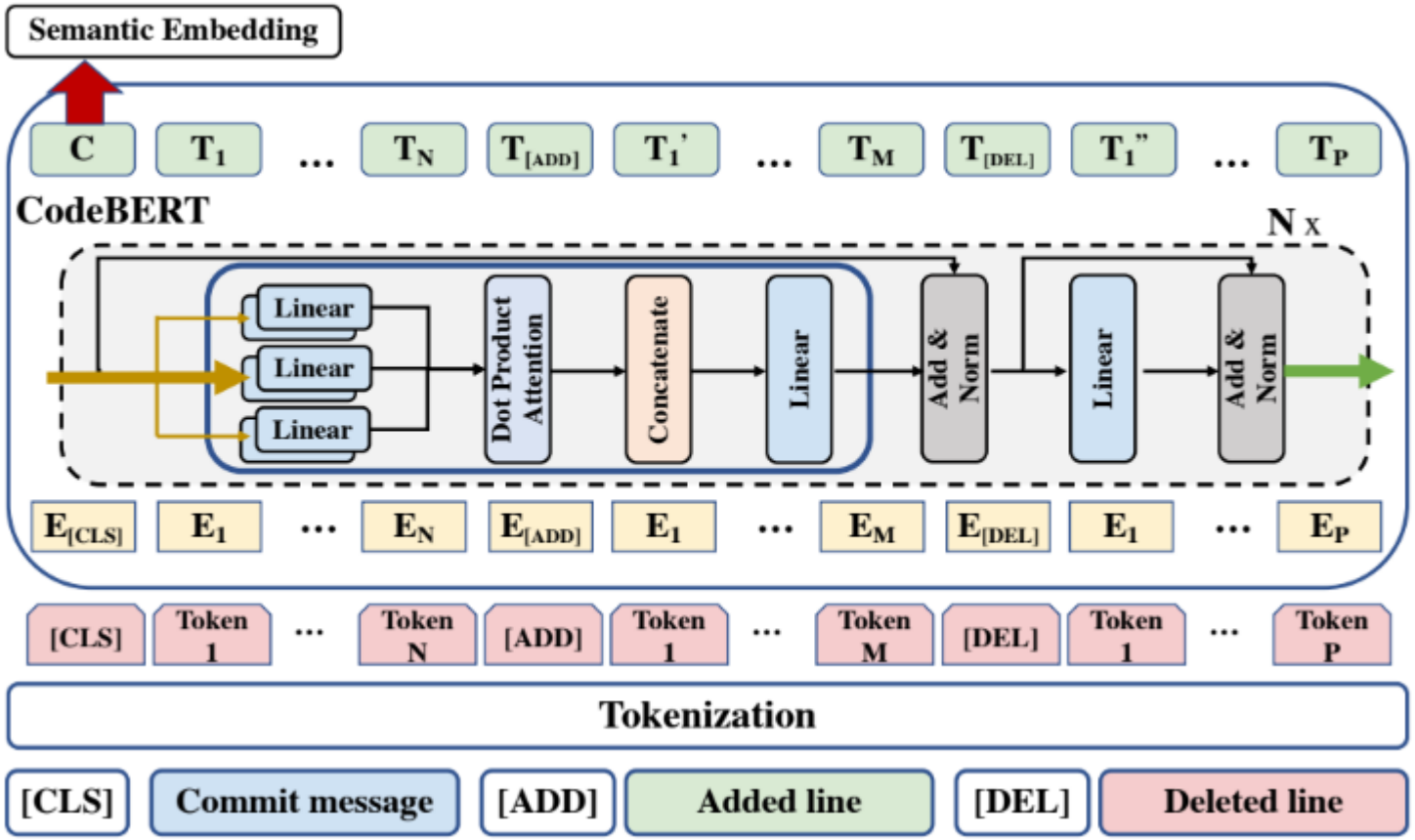
JIT-Fine 算法原理

- 总体框架
  - 特征提取：包括利用CodeBERT生成的嵌入向量和14个专家特征
  - 集成特征学习：利用全连接网络合并训练语义特征和专家特征
  - 缺陷预测和定位：



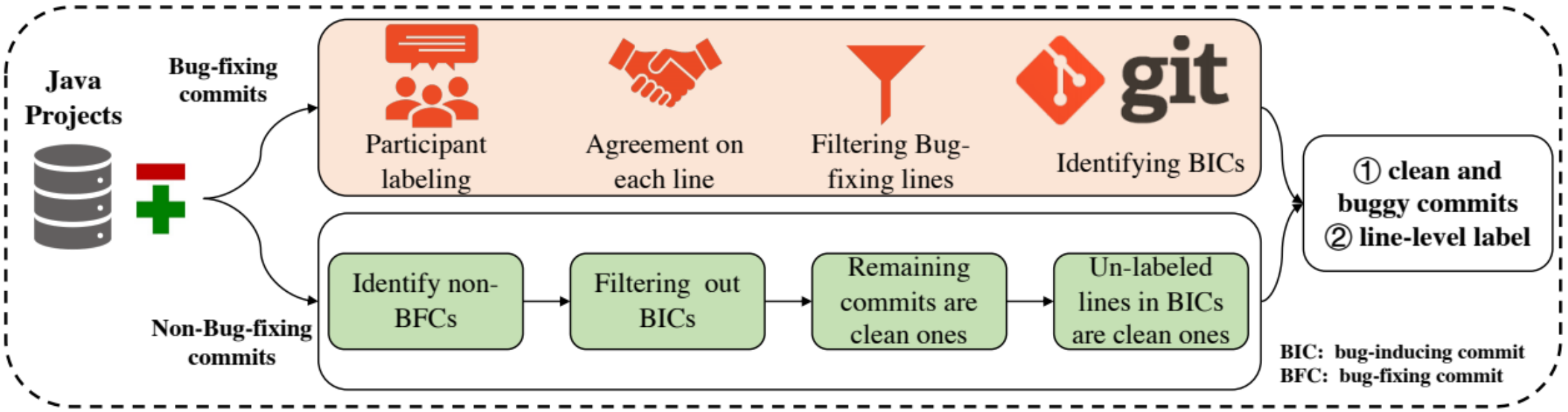
JIT-Fine 算法原理

- 特征提取
  - 将统计数据和代码Tokens转换为数字表示，捕获缺陷预测和定位任务的区别特征
  - 利用CommitGuru提取14个广泛使用的变更级度量作为专家特征
  - 利用CodeBERT提取代码变更的语义特征
- 集成特征学习
  - 将专家特征与语义特征合并



JIT-Fine 算法原理

- 数据集构建
  - 缺陷修复的提交（BIC）：
  - 缺陷未修复的提交（BFC）：



JIT-Fine 实验分析

- 基线
  - LApredict利用逻辑回归分类器和“添加代码行”专家特征
  - Deeper利用深度信念网络学习专家特征
  - CC2Vec利用分层注意力网络学习语义特征
  - Yan利用逻辑回归和N-gram技术建立缺陷定位模型，学习专家特征
  - JITLine利用专家特征和令牌特征的组合构建随机森林缺陷预测模型，利用令牌特征构建缺陷定位模型
  - JIT-Fine利用最佳语义特征和专家特征构建统一的缺陷预测和定位模型

Baselines	Features			Model		JIT Tasks		Venue
	EF	SF	TF	TM	DLM	DP	DL	
LApredict [62]	✔			✔		✔		ISSTA 2021
Deeper [59]	✔				✔	✔		QRS 2015
DeepJIT [17]		✔			✔	✔		MSR 2019
CC2Vec [18]		✔			✔	✔		ICSE 2020
Yan et al. [57]	✔		✔	✔		✔*	✔*	TSE 2020
JITLine [42]	✔		✔	✔		✔*	✔*	MSR 2021
JIT-Fine	✔	✔			✔	✔	✔	This work

“EF”：专家特征
“SF”：语义特征
“TF”：Token特征
“TM”：传统机器学习
“DLM”：深度学习模型

- 评价指标
  - 工作量感知指标（effort-aware）：
    - 开发者根据预测模型的预测结果进行代码审查时，审查一定数量的代码（即工作量）所能检查到的缺陷数量或者比例。在即时缺陷预测工作中，研究者通常将代码审查工作量设置为所有变更修改代码总行的 20%，并且提出使用在检查 20%代码的情况下来计算查准率（Precision@20%）、查全率（Recall@20%）、（F1-measure(F1@20%)）
    - Effort@20%：开发人员在测试数据集中发现20%的实际缺陷引入提交时，开发人员必须花费的工作量
  - P_{opt}表示预测模型召回率和缺陷检测工作量之间的关系

$$1 - \frac{\text{Area}(Optimal) - Area(M)}{Area(Optimal) - Area(Worst)}$$

JIT-Fine 实验分析

- JIT-Fine的即时缺陷预测性能
  - 目标：JIT-Fine利用集成特征进行即时缺陷预测的性能提升程度
  - 实验设计：
    - 对比六种最先进基线模型，采用工作量不可知指标和工作量感知指标
    - 每个项目的提交按时间戳升序排序，构建训练集（前80%）和测试集（后20%）
  - 结果：
    - JIT-Fine在即时缺陷预测方面优于最先进的基线算法
    - 集成语义和专家特征的统一模型可以获得更好的性能

Methods	F1-score↑	AUC↑	R@20%E↑	E@20%R↓	Popt↑
LApredict	0.059	0.694	0.625	0.020	0.814
Yan et al.	0.062	0.675	0.615	0.022	0.819
Deeper	0.246	0.682	0.638	0.021	0.827
DeepJIT	0.293	0.775	0.676	0.014	0.860
CC2Vec	0.248	0.791	0.676	0.014	0.861
JITLine	0.261	0.802	0.705	0.015	0.883
JIT-Fine	**0.431**	**0.881**	**0.773**	**0.010**	**0.927**

JIT-Fine 实验分析

- 集成特征的有效性
  - 目标：语义特征和专家特征集成对即时缺陷预测有效性的影响
  - 实验设计：
    - 三种训练场景，分别利用语义特征SF、专家特征EF和两类特征集成EF+SF
    - 工作量不可知指标和工作量感知指标
  - 结果：
    - 在缺陷预测领域，语义特征比专家特征更能理解代码特征
    - JIT-Fine利用集成特征优势，在两类指标中表现更好

Methods	Setting	F1-score↑	AUC↑	R@20%E↑	E@20%R↓	Popt↑
JITLine	EF	0.147	0.672	0.617	0.024	0.818
	SF_t	0.191	0.783	**0.722**	**0.012**	**0.894**
	$EF+SF_t$	**0.261**	**0.802**	0.705	0.015	0.883
JIT-Fine	EF	0.230	0.661	0.632	0.020	0.825
	SF	0.375	0.856	0.741	0.015	0.917
	EF+SF	**0.431**	**0.881**	**0.773**	**0.010**	**0.927**

JIT-Fine 实验分析

- 即时缺陷定位
  - 目标：JIT-Fine是否可以进行准确的即时缺陷定位
  - 实验设计：
    - 利用PyDriller识别缺陷引入提交，利用git blame获取代码行标签
    - JIT-Fine利用微调的CodeBERT中每个Token的权重计算对分类器的影响
  - 结果：
    - JIT-Fine具有更好的缺陷定位性能
    - 构建具有集成特征的统一模型有利于缺陷预测和缺陷定位任务

Methods	Accuracy↑		R@20%E$_l$↑	E@20%R$_l$↓	IFA$_l$↓
	Top-5	Top-10			
JITLine	0.104	0.098	0.157	0.332	24.2
Yan et al.	0.193	0.195	0.143	0.345	15.3
JIT-Fine	**0.212**	**0.214**	**0.208**	**0.318**	**10.8**

JIT-Fine 优劣分析

- 优势：
  - 充分利用缺陷修复后的源代码专家特征和上下文语义特征，可同时实现即时缺陷预测和缺陷定位
  - 构建了大规模代码行级标记数据集JIT-Defects4J
  - 分析了集成特征在两种即时缺陷预测与定位算法上的有效性

- 局限性
  - 提取并集成了专家特征和语义特征，增加了预处理过程的复杂度

DeepDL

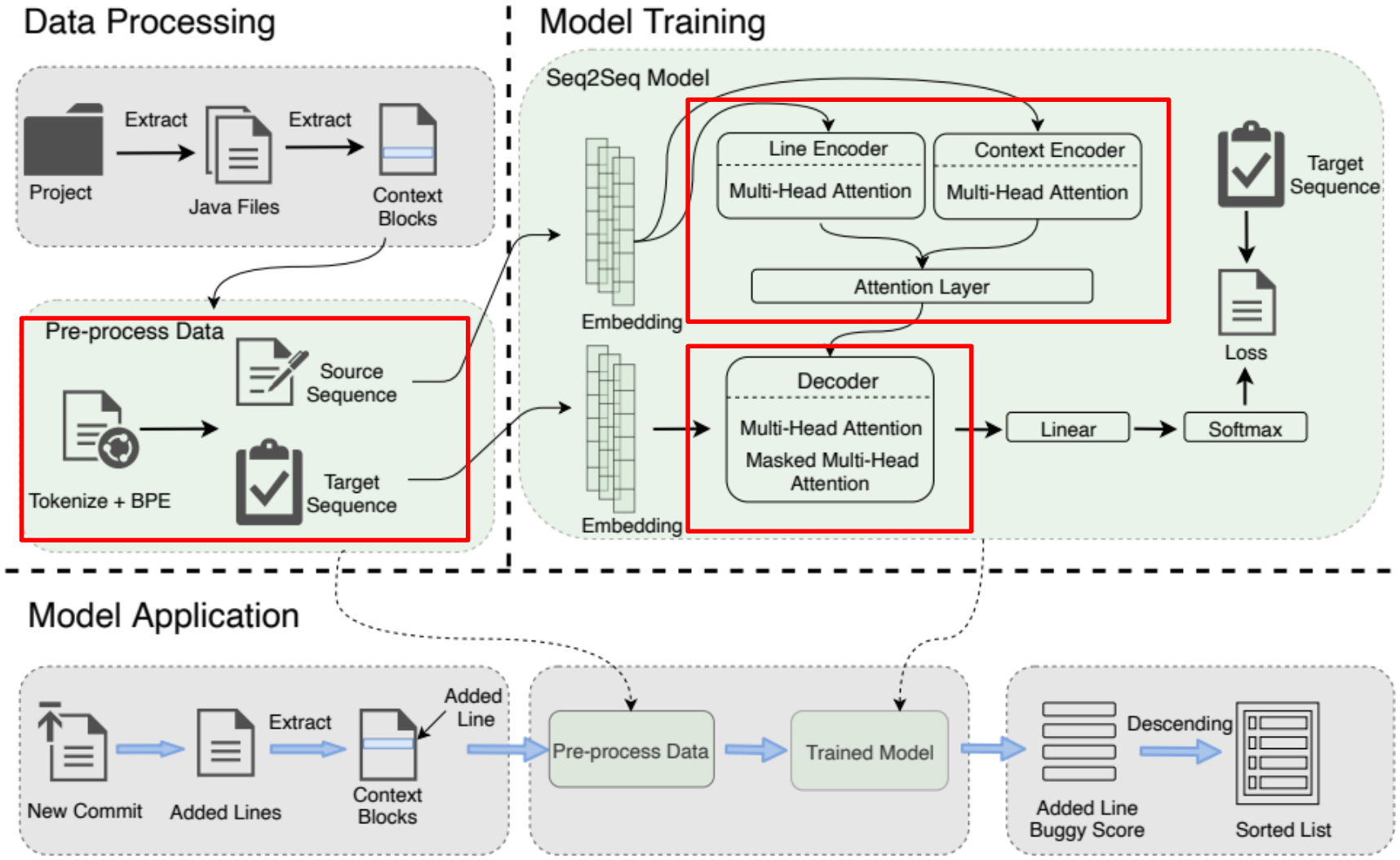
DeepDL

T	目标	即时缺陷预测、定位引入缺陷的代码行
I	输入	软件代码提交
P	处理	1. 从软件项目的最新快照中收集代码行作为训练样本，训练神经语言模型 2. 利用采集的代码行训练DeepDL模型 3. 预测新提交的代码变更缺陷并定位缺陷代码行
O	输出	软件代码变更是否存在缺陷和缺陷引入代码行

P	问题	利用代码行之间的语义特征和上下文关系
C	条件	缺陷引入提交中标记行标签，构建含上下文信息的代码块
D	难点	如何实现高准确率的缺陷引入代码行定位
L	水平	2021 TSE（CCF-A）

DeepDL 算法原理

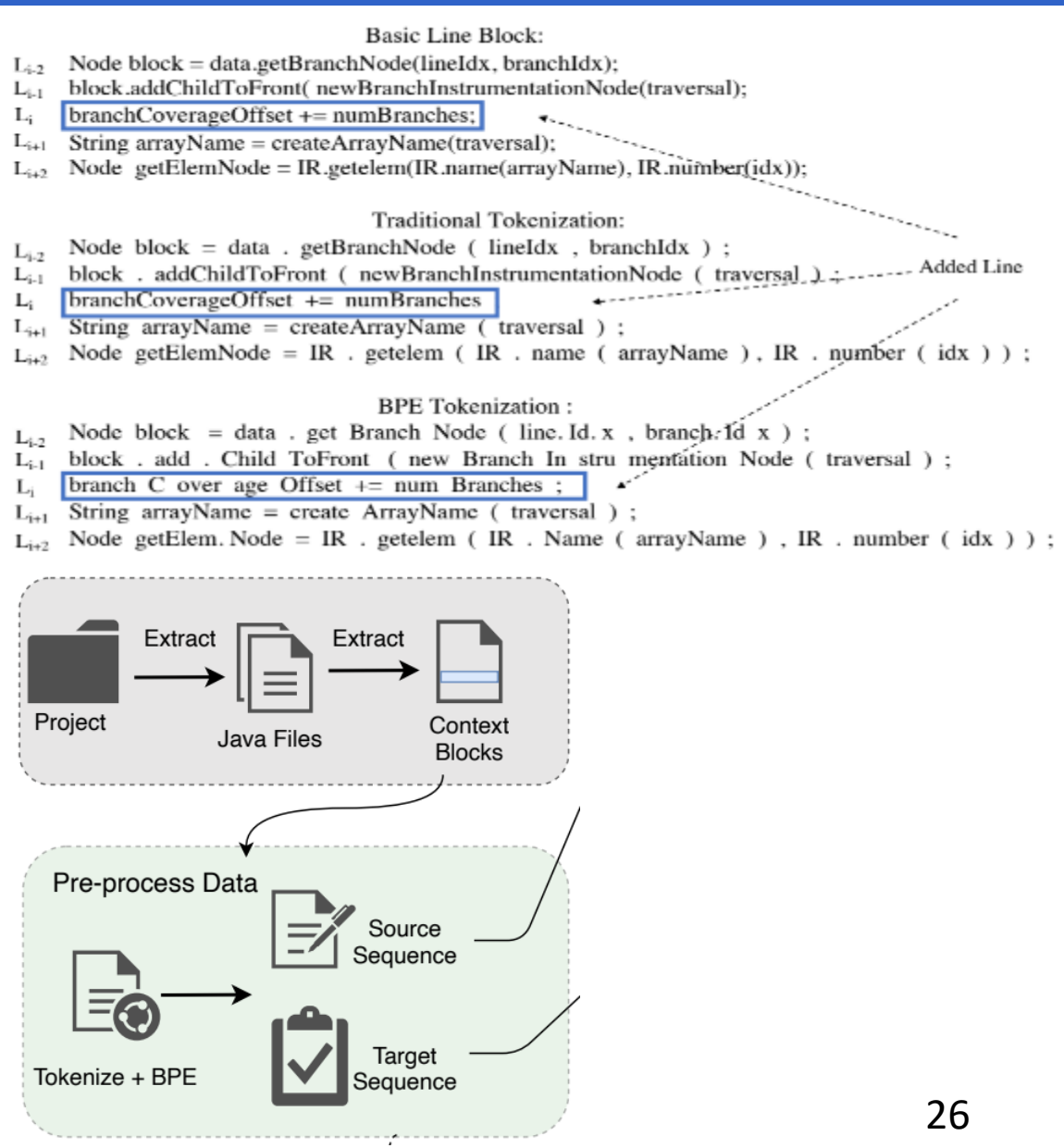
- 总体框架



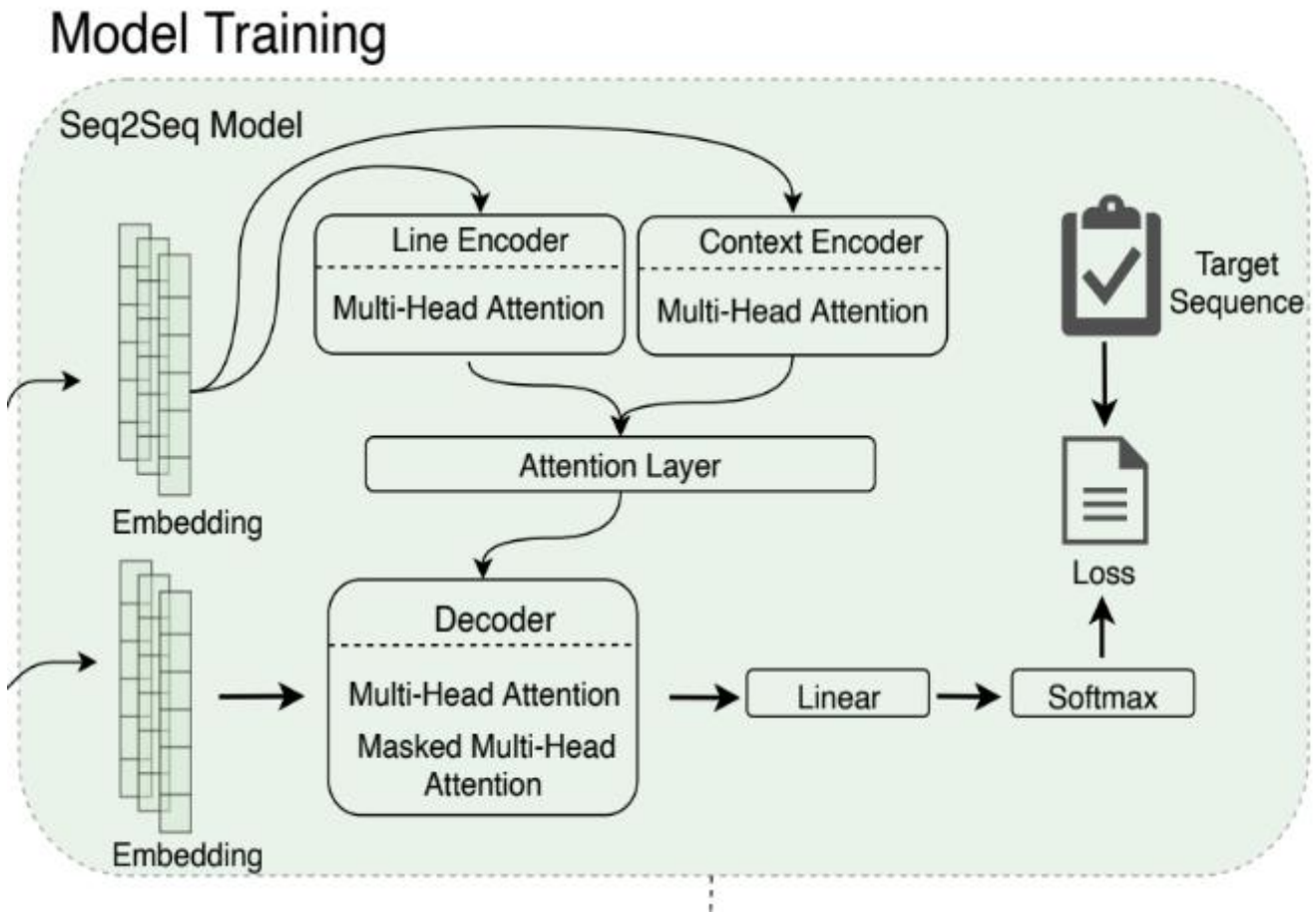
DeepDL 算法原理

- 数据处理
  - 收集训练和测试集
    - 分别识别干净和缺陷代码行
  - 处理训练和测试集
    - 删除了测试代码、空行、导入语句和注释
    - 5个代码行组成一个代码行块

      $[l_{i-2}, l_{i-1}, l_i, l_{l+1}, l_{i+2}]$
  - 分词和构建词表
    - 利用字节对编码（BPE）数据压缩技术迭代采集单词，构建词汇表



- 模型训练
  - 编码器：Seq2Seq模型，从代码行学习潜在特征
  - 解码器：生成目标序列
  - 数据流：$[l_{i-2}, l_{i-1}, l_i, l_{l+1}, l_{i+2}]$ $[l_i]$输入代码行编码器，$[l_{i-2}, l_{i-1}, l_{l+1}, l_{i+2}]$输入上下文编码器
  - 解码器：生成目标序列
  - 损失函数：交叉熵



DeepDL 实验分析

- 有效性评估
  - CC2Vec模型在不同的评估指标下性能较差
  - DeepDL在所有评估指标下平均性能优于Yan的方法
  - 在所有14个项目中， DeepDL模型的改进效果显著

Project	Top-1 accuracy			Top-5 accuracy			MRR			MAP		
	Yan's	CC2Vec	Ours	Yan's	CC2Vec	Ours	Yan's	CC2Vec	Ours	Yan's	CC2Vec	Ours
Activemq	0.3936	0.3417	**0.4070**	0.5628	0.5142	**0.6348**	0.4785	0.4317	**0.5117**	0.4505	0.4217	**0.4749**
Closure-compiler	0.2432	0.2226	**0.2774**	0.4966	0.4452	**0.4983**	0.3650	0.3404	**0.3933**	0.3504	0.3289	**0.3718**
Deeplearning4j	0.2451	0.1963	**0.3184**	0.4971	0.4189	**0.5908**	0.3702	0.3150	**0.4464**	0.3264	0.2847	**0.3746**
Druid	**0.2107**	0.1517	0.1966	0.4157	0.2949	**0.4663**	0.3149	0.2330	**0.3297**	0.2720	0.2043	**0.2975**
Flink	0.2065	0.1716	**0.2445**	0.4146	0.3470	**0.4761**	0.3125	0.2625	**0.3584**	0.2608	0.2307	**0.2894**
Graylog2-server	0.3384	0.3207	**0.3742**	0.5951	0.5249	**0.6731**	0.4637	0.4207	**0.5077**	0.4222	0.3961	**0.4503**
Jenkins	0.3602	0.2951	**0.3748**	0.6184	0.5436	**0.6699**	0.4834	0.4149	**0.5149**	0.4536	0.4026	**0.4792**
Jetty	0.2452	0.1928	**0.2773**	0.4702	0.4279	**0.5427**	0.3550	0.3104	**0.4022**	0.3210	0.2838	**0.3537**
Jitsi	0.3475	0.3126	**0.3634**	0.6297	0.5849	**0.6798**	0.4798	0.4373	**0.5043**	0.4325	0.4007	**0.4425**
Jmeter	0.4545	0.4103	**0.4980**	0.7462	0.6743	**0.7968**	0.5838	0.5336	**0.6285**	0.5615	0.5188	**0.6048**
Libgdx	0.3448	0.2808	**0.3711**	0.6010	0.5107	**0.6568**	0.4668	0.3945	**0.4981**	0.4258	0.3740	**0.4582**
Robolectric	0.2368	0.1699	**0.2536**	0.4928	0.4115	**0.5383**	0.3654	0.2920	**0.3851**	0.3145	0.2662	**0.3301**
Storm	0.0971	0.0874	**0.1748**	0.3495	0.2524	**0.3689**	0.2062	0.1840	**0.2789**	0.1810	**0.2338**	**0.2338**
H2o	0.2584	0.2299	**0.3057**	0.5258	0.4483	**0.5954**	0.3854	0.3427	**0.4427**	0.3542	0.3197	**0.3996**
average	*0.2844*	*0.2416*	***0.3169***	*0.5297*	*0.4571*	***0.5849***	0.4022	*0.3509*	***0.4430***	*0.3662*	*0.3197*	***0.3972***
p-value	*<0.001*	<0.001		*<0.001*	*<0.001*		*<0.001*	*<0.001*		*<0.001*	*<0.001*	

DeepDL 实验分析

- 跨项目评估
  - 采用跨项目数据训练DeepDL可行，可以实现与项目内缺陷定位类似的性能

Project	Top-1 accuracy			Top-5 accuracy			MRR			MAP		
	CP	WP	Improve	CP	WP	Improve	CP	WP	Improve	CP	WP	Improve
Activemq	**0.4154**	0.4070	-2.01%	0.6181	**0.6348**	2.71%	**0.5146**	0.5117	-0.57%	**0.4779**	0.4749	-0.63%
Closure-compiler	0.2759	**0.2774**	0.54%	**0.5017**	0.4983	-0.68%	**0.3962**	0.3933	-0.73%	**0.3743**	0.3718	-0.66%
Deeplearning4j	0.3174	**0.3184**	0.30%	0.5879	**0.5908**	0.50%	0.4442	**0.4464**	0.50%	0.3720	**0.3746**	0.71%
Druid	**0.2022**	0.1966	-2.76%	0.4579	**0.4663**	1.83%	0.3249	**0.3297**	1.48%	0.2896	**0.2975**	2.74%
Flink	**0.2498**	0.2445	-2.12%	**0.4951**	0.4761	-3.84%	**0.3640**	0.3584	-1.54%	**0.2900**	0.2894	-0.21%
Graylog2-server	**0.3908**	0.3742	-4.25%	**0.6731**	0.6731	-0.01%	**0.5169**	0.5077	-1.78%	**0.4536**	0.4503	-0.73%
Jenkins	0.3592	**0.3748**	4.33%	0.6573	**0.6699**	1.92%	0.5002	**0.5149**	2.94%	0.4670	**0.4792**	2.62%
Jetty	0.2617	**0.2773**	5.97%	0.5298	**0.5427**	2.43%	0.3870	**0.4022**	3.93%	0.3434	**0.3537**	3.00%
Jitsi	**0.3695**	0.3634	-1.64%	0.6737	**0.6798**	0.91%	**0.5068**	0.5043	-0.50%	**0.4428**	0.4425	-0.08%
Jmeter	0.4830	**0.4980**	3.11%	0.7794	**0.7968**	2.24%	0.6146	**0.6285**	2.26%	0.5946	**0.6048**	1.71%
Libgdx	**0.3744**	0.3711	-0.88%	0.6355	**0.6568**	3.35%	**0.4982**	0.4981	-0.02%	0.4565	**0.4582**	0.38%
Robolectric	0.2512	**0.2536**	0.95%	**0.5526**	0.5383	-2.59%	**0.3856**	0.3851	-0.13%	**0.3361**	0.3301	-1.77%
Storm	0.1650	**0.1748**	5.91%	**0.4078**	0.3689	-9.53%	0.2758	**0.2789**	1.11%	**0.2341**	0.2338	-0.13%
H2o	**0.3128**	0.3057	-2.27%	0.5865	**0.5954**	1.52%	**0.4453**	0.4427	-0.58%	0.3975	**0.3996**	0.53%
average	*0.3163*	***0.3169***	*0.19%*	*0.5826*	***0.5849***	*0.39%*	*0.4410*	***0.4430***	*0.45%*	*0.3950*	***0.3972***	*0.56%*
p-value	*>0.05*			*<0.05*			*<0.001*			*<0.001*		

CP：跨项目缺陷定位　　WP：项目内缺陷定位

- 消融评估
  - WB（Without BP）模型：在数据处理阶段使用传统的标记方法代替BPE标记技术

Project	Top-1 accuracy			Top-5 accuracy			MRR			MAP		
	WB	Original	Improve	WB	Original	Improve	WB	Original	Improve	WB	Original	Improve
Activemq	0.3735	**0.4070**	8.97%	0.5796	**0.6348**	9.54%	0.4762	**0.5117**	7.44%	0.4580	**0.4749**	3.68%
Closure-compiler	**0.2894**	0.2774	-4.14%	**0.5086**	0.4983	-2.02%	**0.3960**	0.3933	-0.67%	0.3701	**0.3718**	0.47%
Deeplearning4j	0.2988	**0.3184**	6.54%	0.5518	**0.5908**	7.08%	0.4215	**0.4464**	5.91%	0.3546	**0.3746**	5.65%
Druid	**0.2360**	0.1966	-16.67%	0.4522	**0.4663**	3.11%	**0.3449**	0.3297	-4.42%	0.2968	**0.2975**	0.26%
Flink	0.2103	**0.2445**	16.25%	0.4366	**0.4761**	9.04%	0.3250	**0.3584**	10.28%	0.2668	**0.2894**	8.47%
Graylog2-server	0.3729	**0.3742**	0.34%	0.6054	**0.6731**	11.18%	0.4843	**0.5077**	4.84%	0.4314	**0.4503**	4.38%
Jenkins	0.3670	**0.3748**	2.12%	0.6379	**0.6699**	5.02%	0.4950	**0.5149**	4.02%	0.4517	**0.4792**	6.09%
Jetty	0.2525	**0.2773**	9.82%	0.4995	**0.5427**	8.64%	0.3756	**0.4022**	7.09%	0.3272	**0.3537**	8.10%
Jitsi	0.3422	**0.3634**	6.21%	0.6351	**0.6798**	7.05%	0.4798	**0.5043**	5.09%	0.4211	**0.4425**	5.07%
Jmeter	0.4957	**0.4980**	0.48%	0.7897	**0.7968**	0.90%	0.6285	**0.6285**	0.00%	0.5992	**0.6048**	0.93%
Libgdx	0.3415	**0.3711**	8.65%	0.6190	**0.6568**	6.10%	0.4647	**0.4981**	7.19%	0.4238	**0.4582**	8.13%
Robolectric	0.2297	**0.2536**	10.42%	0.4785	**0.5383**	12.50%	0.3529	**0.3851**	9.13%	0.3089	**0.3301**	6.87%
Storm	0.1650	**0.1748**	5.88%	**0.3883**	0.3689	-5.00%	0.2726	**0.2789**	2.30%	0.2137	**0.2338**	9.42%
H2o	0.2941	**0.3057**	3.94%	0.5793	**0.5954**	2.78%	0.4306	**0.4427**	2.81%	0.3814	**0.3996**	4.77%
average	*0.3049*	***0.3169***	*3.94%*	*0.5544*	***0.5849***	*5.50%*	*0.4248*	***0.4430***	*4.27%*	*0.3789*	***0.3972***	*4.82%*
p-value	*<0.001*			*<0.001*			*<0.001*			*<0.001*		

DeepDL 实验分析

- 消融评估
  - WC（Without Context）模型：删除了添加到代码行中的上下文信息，改为使用单一代码行训练DeepDL

Project	Top-1 accuracy			Top-5 accuracy			MRR			MAP		
	WC	Original	Improve	WC	Original	Improve	WC	Original	Improve	WC	Original	Improve
Activemq	0.3819	**0.4070**	6.58%	0.6147	**0.6348**	3.27%	0.4930	**0.5117**	3.79%	0.4612	**0.4749**	2.96%
Closure-compiler	0.2723	**0.2774**	1.89%	**0.5188**	0.4983	-3.96%	0.3932	**0.3933**	0.04%	0.3687	**0.3718**	0.85%
Deeplearning4j	0.2734	**0.3184**	16.43%	0.5566	**0.5908**	6.14%	0.4100	**0.4464**	8.88%	0.3588	**0.3746**	4.41%
Druid	0.1798	**0.1966**	9.38%	0.4466	**0.4663**	4.40%	0.3101	**0.3297**	6.32%	0.2837	**0.2975**	4.89%
Flink	0.2422	**0.2445**	0.94%	0.4655	**0.4761**	2.28%	0.3515	**0.3584**	1.95%	0.2866	**0.2894**	0.98%
Graylog2-server	0.3678	**0.3742**	1.74%	**0.6807**	0.6731	-1.13%	0.5058	**0.5077**	0.37%	**0.4541**	0.4503	-0.84%
Jenkins	0.3699	**0.3748**	1.31%	0.6563	**0.6699**	2.07%	0.5064	**0.5149**	1.67%	0.4714	**0.4792**	1.66%
Jetty	0.2498	**0.2773**	11.03%	0.5271	**0.5427**	2.96%	0.3797	**0.4022**	5.94%	0.3364	**0.3537**	5.12%
Jitsi	**0.3680**	0.3634	-1.24%	0.6715	**0.6798**	1.24%	**0.5073**	0.5043	-0.60%	**0.4451**	0.4425	-0.60%
Jmeter	0.4964	**0.4980**	0.32%	0.7834	**0.7968**	1.72%	0.6241	**0.6285**	0.71%	0.5986	**0.6048**	1.03%
Libgdx	0.3662	**0.3711**	1.35%	0.6289	**0.6568**	4.44%	0.4898	**0.4981**	1.70%	0.4487	**0.4582**	2.12%
Robolectric	0.2368	**0.2536**	7.07%	0.5144	**0.5383**	4.65%	0.3693	**0.3851**	4.27%	0.3247	**0.3301**	1.66%
Storm	**0.1845**	0.1748	-5.26%	**0.4466**	0.3689	-17.39%	**0.2988**	0.2789	-6.68%	**0.2382**	0.2338	-1.84%
H2o	0.2825	**0.3057**	8.20%	0.5517	**0.5954**	7.92%	0.4143	**0.4427**	6.86%	0.3770	**0.3996**	5.99%
average	*0.3068*	***0.3169***	*3.28%*	*0.5778*	***0.5849***	*1.23%*	*0.4338*	***0.4430***	*2.13%*	*0.3905*	***0.3972***	*1.71%*
p-value	*<0.001*			*<0.001*			*<0.001*			*<0.001*		

DeepDL 优劣分析

- 优势：
  - 利用BFE算法标记源代码，可以大幅降低源代码词汇表大小，缓解Out-of-Vocabulary问题
  - 利用神经语言模型学习代码行的自然性，有效捕捉代码行上下文特征及其关系
  - 提高跨项目缺陷定位准确率，与项目内缺陷定位性能相当

- 局限性
  - 在项目的缺陷引入变更较少的情况下，DeepDL预测精度不佳

总结

- 应用总结
  - 即时缺陷预测在工程实践中进行大规模推广仍然存在以下挑战：
    - 缺乏准确的数据环境
    - 缺乏多维的数据特征
    - 缺乏统一的评估方法
- 前沿发展
  - 加强噪音数据的处理，提出更准确的数据标注方法
  - 综合考虑多源软件制品，提取多维特征
  - 研究更先进的建模技术，取得建模技术上的突破

参考文献

- Ni C, Wang W, Yang K, et al. The best of both worlds: integrating semantic features with expert features for defect prediction and localization[C]//Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2022: 672-683.
- Qiu F, Gao Z, Xia X, et al. Deep Just-In-Time Defect Localization[J]. IEEE Transactions on Software Engineering, 2021.
- Spadini D, Aniche M, Bacchelli A. Pydriller: Python framework for mining software repositories[C]//Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2018: 908-911.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！