

Self-Admitted Technical Debtsの検出と説明のための Attentionベースニューラルネットワーク

Detecting and Explaining Self-Admitted Technical Debts with Attention-based Neural Networks

Xin Wang
School of Computer Science, Wuhan
University
Wuhan, China
xinwang0920@whu.edu.cn

Jin Liu*
School of Computer Science, Wuhan
University
Wuhan, China
jinliu@whu.edu.cn

Li Li
Faculty of Information Technology,
Monash University
Melbourne, Australia
Li.Li@monash.edu

Xiao Chen
Faculty of Information Technology,
Monash University
Melbourne, Australia
xiao.chen@monash.edu

Xiao Liu
School of Information Technology,
Deakin University
Geelong, Australia
xiao.liu@deakin.edu.au

Hao Wu
School of Information Science and
Engineering, Yunnan University
Kunming, China
haowu@ynu.edu.cn

論文情報：

Xin Wang, Jin Liu, Li Li, Xiao Chen, Xiao Liu, and Hao Wu. 2020. Detecting and explaining self-admitted technical debts with attention-based neural networks.

In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20). Association for Computing Machinery, New York, NY, USA, 871–882.

紹介者：堀口

技術的負債 (Technical Debt)

開発者が迅速にソフトウェアをリリースするために、

「**時間はかかるが最適な方法**」ではなく、

「**今すぐできるが最適ではない方法**」を採用することで、
後から変更作業が必要になる設計やコード

典型例：

- ・ ハードコード
- ・ 他モジュールからのコードのコピー
- ・ テストの失敗を無視したコード変更

Self-Admitted Technical Debt (SATD)

開発者が意図的に導入した技術的負債をドキュメント化したもの

具体例：

例：JMeterプロジェクトのとあるコードコメント

「May be replaced later（後で置き換える可能性がある）」は、
現在のコードが一時的な解決策で、将来的に置き換える必要がある

※本論文では、SATDのコードコメントをSATDと呼称

技術的負債の蓄積による問題

負債を返済しない状態が長期に渡ると、ソフトウェアの保守に悪影響^[18, 21, 23, 27, 35, 53]

⇒ 技術的負債の早期検出，修正を自動化する手法が求められる

先行研究

SATDの検出手法を提案^[16, 38]

(SATDは開発者が認知し，ドキュメント化されているため分析が容易)

→ 専門家によるテキストマイニング，パターン抽出手法が主なアプローチ

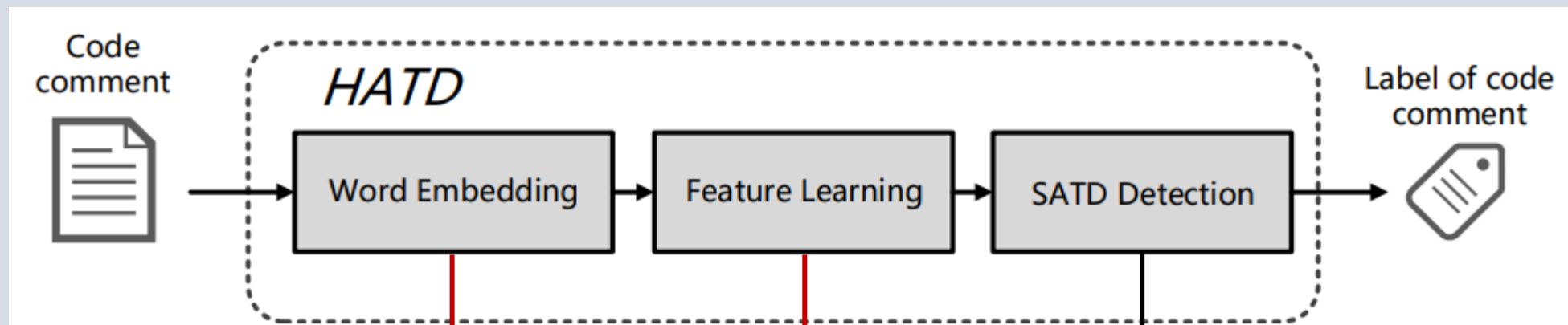
- 汎用性が低い
- 手動での特徴抽出が必要

目的&アプローチ

目的

汎用性が高いAttentionベースのSATD検出モデルを構築

アプローチ



1. 文脈を考慮した単語埋め込み

2. AttentionベースのNNによる特徴表現学習

3. 特徴表現をもとにSATD, non-SATDを分類

ELMo^[37]

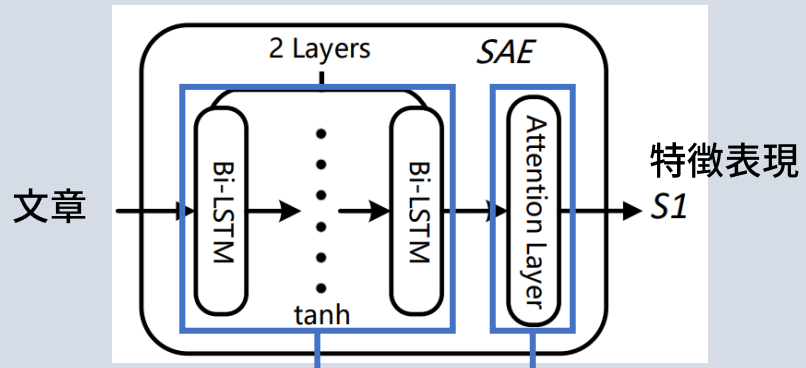
- 文章を入力とし，Bi-LSTMを用いて文脈に応じた単語ベクトルを出力するモデル
- 大規模コーパスで事前学習したモデルを，
自前のモデルの中間層として統合，個別タスクに合わせた転移学習
- 採用理由
例：「rework」
直訳：やり直す
SATDを示しそうな単語だが，あるプロジェクトでは定期的なリファクタリングのことを指していて，必ずしもSATDを示していない
→ 文脈に応じて異なる埋め込み表現を使えば，ある程度識別可能

AttentionベースのNNによる特徴表現学習

概要

- ・ 説明可能なモデルを構築するための，Bi-LSTM with Attention^[64]モデル
 - ・ 単語間の複雑な関係性を抽出できる，Transformer-based^[48]モデル
- ⇒ 2つの出力を結合して特徴表現を学習

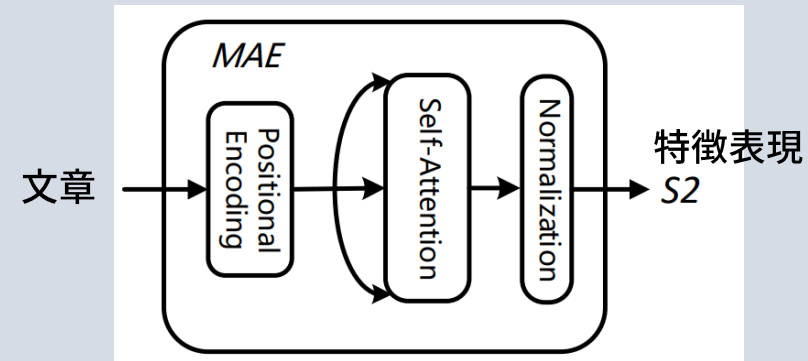
Bi-LSTM with Attention



文脈を学習

SATD検出に重要な単語，フレーズを学習

Transformer-based

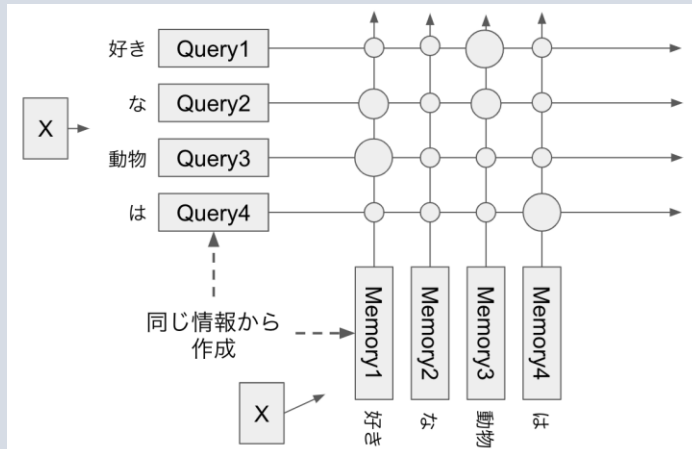


次のスライドで説明

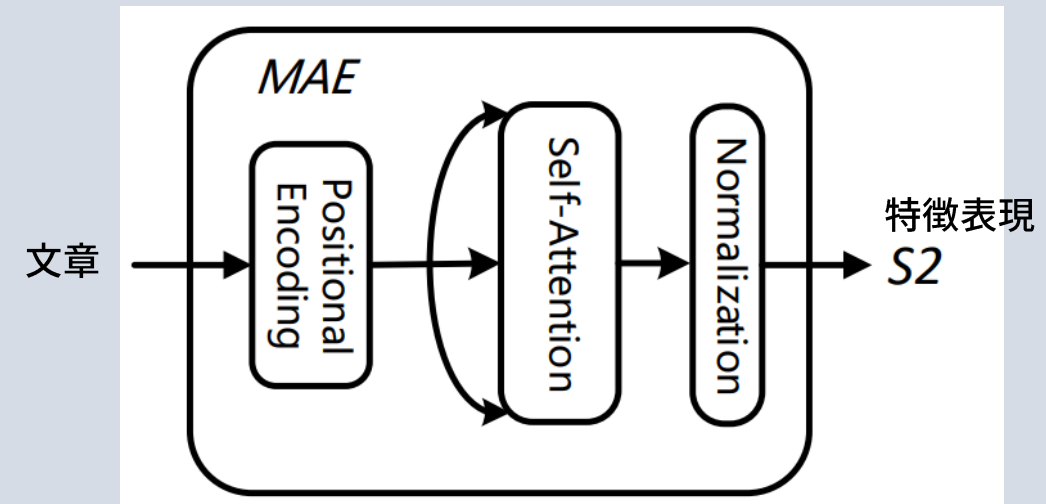
AttentionベースのNNによる特徴表現学習

Transformer-based

- BERTの一部．大量の文章を高速に学習するための手法
- Positional Encoding
単語ベクトルに位置情報を埋め込む
→ RNNを使わずとも文脈の学習が可能
- Self-Attention



画像出典：<https://qiita.com/halhorn/items/c91497522be27bde17ce>



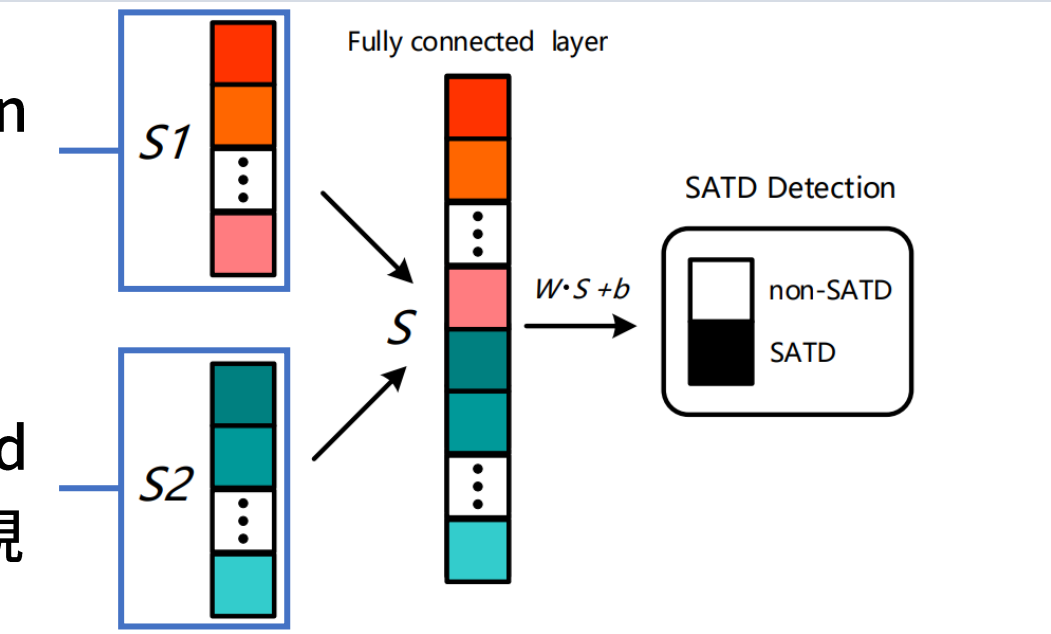
文章中的ある単語を，ほかの単語との
関連性（Attentionの重み）によって特徴づける
→ 単語間の複雑な関係性を抽出できる

特徴表現をもとにSATD，non-SATDを分類

全結合

Bi-LSTM with Attention
で学習した特徴表現

Transformer-based
で学習した特徴表現



リサーチクエスチョン

提案手法を評価するために以下のRQに回答

- RQ1：提案手法はSATD検出に効果的か？
- RQ2：先行研究の手法よりも優れているか？
- RQ3：ELMoを使った単語埋め込みは他の単語埋め込み手法と比べて優れているか？
- RQ4：実際のOSSプロジェクトでSATDを検出できるか？

データセット

- 既存研究^[7]のデータセットを利用
- 10件のOSSプロジェクト
SATDコメント：411件，non-SATDコメント：6,257件

モデルの学習方法

2つの方法で学習，比較

- **Within-Project**：プロジェクトごとに学習，10分割交差検証
- **Cross-Projects**：他の9つのプロジェクトで学習，残りの1プロジェクトで評価

RQ1：提案手法はSATD検出に効果的か？

Project	Within-Project			Cross-Projects		
	Precision	Recall	F1	Precision	Recall	F1
Apache Ant	0.7050	0.6108	0.6545	0.6569	0.7801	0.7132
ArgoUML	0.8731	0.9469	0.9085	0.8615	0.9487	0.9030
Columba	0.9434	0.9616	0.9524	0.8906	0.9598	0.9239
EMF	0.7475	0.8586	0.7992	0.6769	0.8787	0.7647
Hibernate	0.8866	0.9536	0.9189	0.8699	0.9305	0.8992
JEdit	0.5468	0.6554	0.5962	0.7783	0.8797	0.8259
JFreeChart	0.9268	0.8747	0.9000	0.6897	0.7135	0.7014
JMeter	0.9249	0.8938	0.9091	0.8178	0.8715	0.8438
JRuby	0.9568	0.9388	0.9477	0.8977	0.9397	0.9182
Squirrel	0.7978	0.8769	0.8355	0.7814	0.8318	0.8058
Average	0.8309	0.8571	0.8422	0.7921	0.8734	0.8299

- 平均するとよく検出できている
- WithinとCrossの結果を比較すると有意な差はない
⇒ 未知のOSSプロジェクトのSATDでも十分検出できる

RQ2：先行研究の手法よりも優れているか？

- 先行研究の中では最も性能が高い手法^[39]（Attentionなしの深層学習モデル）

Projects	With-Project			Cross-Project		
	Ren's	Ours	Gains	Ren's	Ours	Gains
Apache Ant	0.5977	0.6545	+9.50%	0.6260	0.7132	+13.93%
ArgoUML	0.8895	0.9085	+2.14%	0.8409	0.9030	+7.38%
Columba	0.8545	0.9524	+11.46%	0.7782	0.9239	+18.72%
EMF	0.7010	0.7992	+14.01%	0.5797	0.7647	+31.91%
Hibernate	0.8465	0.9189	+8.55%	0.8035	0.8992	+11.91%
JEdit	0.5564	0.5962	+7.15%	0.5760	0.8259	+43.39%
JFreeChart	0.7759	0.9000	+15.99%	0.6504	0.7014	+7.84%
JMeter	0.8858	0.9091	+2.63%	0.7726	0.8438	+9.22%
JRuby	0.8626	0.9477	+9.87%	0.8445	0.9182	+8.73%
Squirrel	0.7467	0.8355	+11.89%	0.7069	0.8058	+13.99%
Average	0.7717	0.8422	+9.14%	0.7179	0.8299	+15.61%

- Within：F1が2~16%向上
 - Cross：F1が7~43%向上
- 特にCross-Projectsで性能向上

⇒ 既存手法では学習できないような、
他のプロジェクトからのSATD検出に有用な情報を学習可能

RQ3：ELMoを使った単語埋め込みは他の単語埋め込み手法と比べて優れているか？

- 比較対象
 - One-Hot, Word2Vec, fastText：文脈考慮なし
 - GloVe：文脈考慮
 - ELMo：文脈考慮，モデルに統合して転移学習

Projects	Within-Project					Cross-Projects				
	ELMo	One-Hot	Word2Vec	GloVe	FastText	ELMo	One-Hot	Word2Vec	GloVe	FastText
Apache Ant	0.6545	0.6263	0.6242	0.5576	0.5825	0.7132	0.6308	0.6783	0.6226	0.6316
ArgoUML	0.9085	0.9132	0.9059	0.8967	0.8918	0.9030	0.8487	0.8440	0.8290	0.8400
Columba	0.9524	0.9467	0.9024	0.9424	0.9434	0.9239	0.8337	0.8685	0.8469	0.8718
EMF	0.7992	0.7826	0.7273	0.7556	0.7506	0.7647	0.6043	0.6435	0.6376	0.6818
Hibernate	0.9189	0.9126	0.9293	0.8871	0.9592	0.8992	0.8257	0.8331	0.8261	0.8242
JEdit	0.5962	0.5862	0.5714	0.5818	0.5622	0.8259	0.6475	0.6013	0.6118	0.6225
JFreeChart	0.9000	0.8837	0.8947	0.8421	0.8660	0.7014	0.6533	0.6645	0.6095	0.7214
JMeter	0.9091	0.8608	0.8718	0.8741	0.8721	0.8438	0.7451	0.7601	0.7537	0.7659
JRuby	0.9477	0.9000	0.9091	0.8841	0.9524	0.9182	0.8726	0.8548	0.8834	0.8626
Squirrel	0.8355	0.8214	0.7719	0.7970	0.8000	0.8058	0.7070	0.7559	0.7299	0.6872
Average	0.8422	0.8234	0.8108	0.8019	0.8190	0.8299	0.7369	0.7504	0.7351	0.7509

- Crossの場合，ELMoが有意に優れている
→ 他のプロジェクトからSATD検出に有用な単語埋め込みを獲得できる

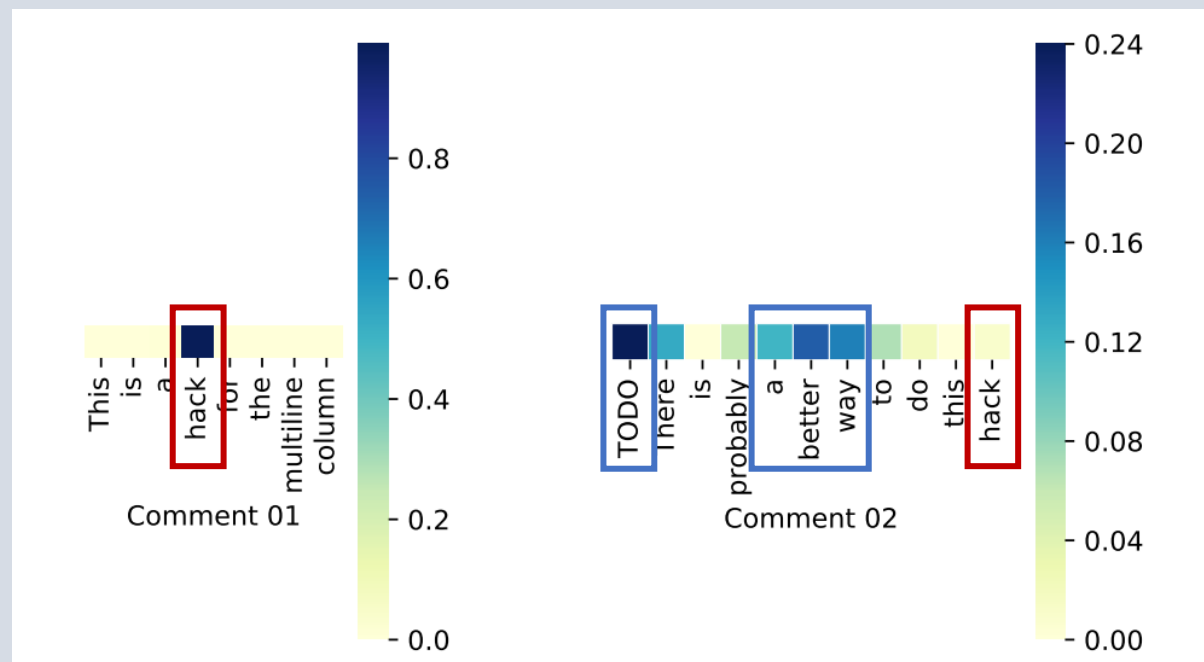
RQ4：実際のOSSプロジェクトでSATDを検出できるか？

- ・ 先述のデータセットを使ってモデルを学習し，
正解データが用意されていない実世界の10個のOSSプロジェクトでSATD検出
- ・ モデルによってSATD，non-SATDと分類されたコメント各50件の計100件を，
5人のプログラマーが目視

False Positive	False Negative
12件	5件

Accuracy：83%

- SATDコメントを入力して，単語ごとのAttentionの重みを確認できる



- 例の2つのサンプルに同じ単語「hack」が含まれているが，重みが大きく違う
→ 同じ単語でも，文脈によって異なる役割を持つ単語を識別できている

目的

汎用性が高いAttentionベースのSATD検出モデルを構築

アプローチ

- ・ 文脈を考慮した単語埋め込み (ELMo) を利用
- ・ 説明可能なモデル (Bi-LSTM with Attention) と
単語間の複雑な関係性を抽出できるモデル (Transformer) を結合

結果

- ・ 既存研究を上回るSATD検出性能，汎化性能を達成
- ・ Attentionの重みによって分類結果の説明が可能であることを確認