

問題解決過程を考慮した不具合報告の 要約生成手法の提案

オープンソースソフトウェア工学研究室
M2 60210045 作島大智

背景

- ソフトウェアプロジェクトでは日々多くの不具合が報告される
 - 開発者は不具合に関する情報を効率的に把握する必要がある



- 開発者は不具合報告の概要理解に時間がかかる
 - 不具合報告の25.9%には15以上のコメント(平均39語)がある [1]
 - 手動で簡素な要約を作成することが推奨されているが、存在するのは2.80%程度 [1]
- 開発者支援を目的として、不具合報告を要約する研究がある

[1] Rastkar, Sarah, Murphy, C Gail, Murray, and Gabriel. 2010.
Summarizing software artifacts: a case study of bug reports. (2010).

背景 | 要約方法

○ 要約方法は2種類存在する

抽出型

対象文の中から重要だと思われる文を抽出する手法

- 文法的におかしい要約文になることはない
- 文中にない単語を利用できないため、読みにくい傾向がある

抽象型

文章の意味を汲み取った上、適切な要約を作成する手法

- 文中にない単語を利用できるため、自由度が高い
- 自然な文章を作成することが困難

○ 不具合報告の要約生成手法は教師あり or 教師なしの機械学習で実装

教師あり [2, 3]

モデル学習のための学習データが必要

- 正解集合を作成するために膨大な手作業が必要
- 一定の専門知識を持つアノテーターが必要
- 精度が学習コーパスの品質に依存

[2] He Jiang, Jingxuan Zhang, Hongjing Ma, Najam Nazam, and Zhilei Ren. 2017. Mining authorship characteristics in bug repositories. *Science China Information Sciences* 60, 1 (2017), 012107.
[3] Sarah Rastkar, Gail C Murphy, and Gabriel Murray. 2014. Automatic summarization of bug reports. *IEEE Transactions on Software Engineering* 40, 4 (2014), 366–380.
[4] Xiaochen Li, He Jiang, Dong Liu, Zhilei Ren, and Ge Li. 2018. Unsupervised deep bug report summarization. In *Proceedings of the 26th Conference on Program Comprehension*. ACM, 144–155.
[5] Qiaozhu Mei, Jian Guo, and Dragomir Radev. 2010. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. Acm, 1009–1018.
[6] Xiaojin Zhu, Andrew Goldberg, Jurgen Van Gael, and David Andrzejewski. 2007. Improving diversity in ranking using absorbing random walks. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 97–104

教師なし [4, 5, 6]

学習データは必要ないが、要約文が冗長になる傾向

→ 代表的な手法：単語の出現頻度に依存する

○ 本研究は教師なし学習を用いた抽出型要約生成手法に着目

DeepSum [7]

- 不具合報告の頻出単語をもとに要約選択を行う
- 以下に考慮したモデルを構築
 - > 参加者のコメントよりも報告者のコメントを優先
 - > ソースコード等の非自然言語より自然言語を優先
- 頻出単語に基づいた要約生成手法は要約文が冗長である傾向

BugSum [8]

- 頻出単語に依存せず、信憑性のある文の選択が高品質な要約生成には必要
 - > テキスト特徴量：オートエンコーダを用いて高次元ベクトルで表現
 - > 信憑性：他の評価コメントが肯定的か否定的かで重みを変化

[7] Xiaochen Li, He Jiang, Dong Liu, Zhilei Ren, and Ge Li. 2018. Unsupervised deep bug report summarization. In Proceedings of the 26th Conference on Program Comprehension. ACM, 144-155.

[8] Haoran Liu, Yue Yu, Shanshan Li, Yong Guo, Deze Wang, and Xiaoguang Mao. 2020.

BugSum: Deep Context Understanding for Bug Report Summarization.

In Proceedings of the 28th International Conference on Program Comprehension (ICPC '20).

Association for Computing Machinery, New York, NY, USA, 94-105. DOI:<https://doi.org/10.1145/3387904.3389272>

先行研究 | 課題点

- 不具合の問題点や要因，解決策を考慮していない
- 要約は報告された不具合の概要理解を効率よく行うために必要
 - 高品質な要約(概要を理解しやすい要約)は以下の要素が必要
 - 報告者が抱える問題点
 - 不具合が生じた要因
 - 不具合の解決方法

着眼点

問題解決過程を含む文章に重みを付与することで、
高品質な要約を選択しやすくする

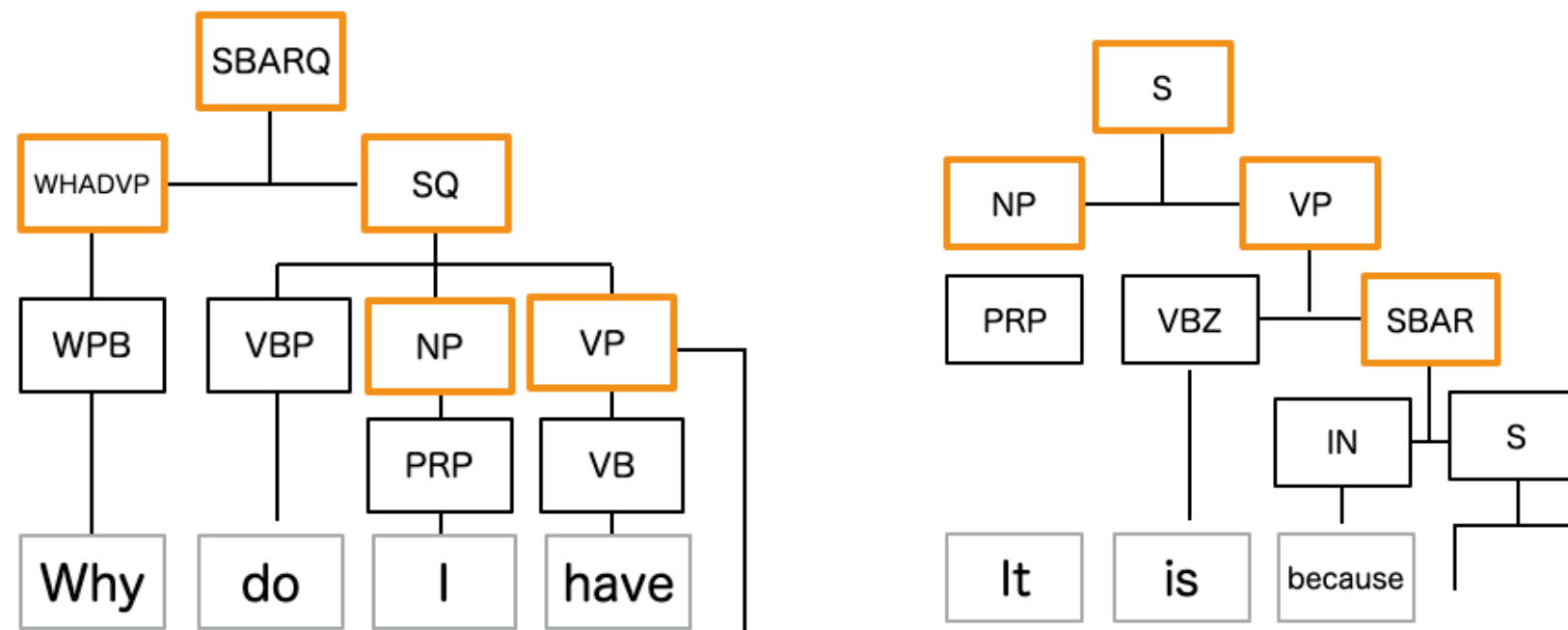
本研究 | 動機

動機

問題解決過程を含む文を抽出して要約文の理解性を向上したい

着眼点

問題, 要因, 解決に該当する文にはそれぞれ特徴的な文構造が存在する



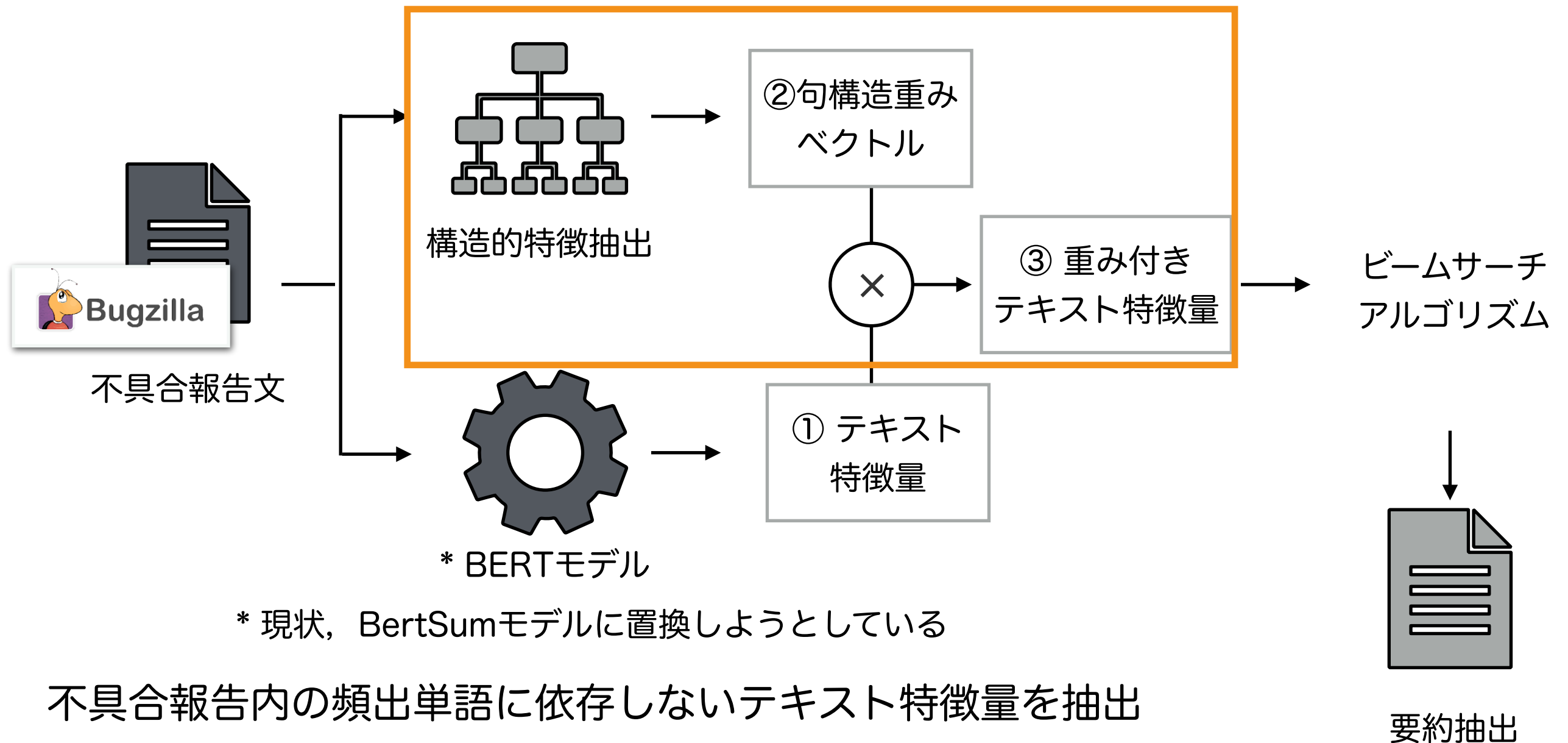
アイデア

問題, 要因, 解決に該当する文の構造が異なるという特徴を利用して各文に重みを付与することで, 問題解決過程を含む文を抽出する精度を上げる

本研究 | 全体図

本研究のアプローチ

- 構造的特徴に応じた各文の重み付け



* 現状, BertSumモデルに置換しようとしている

不具合報告内の頻出単語に依存しないテキスト特徴量を抽出
→ 冗長な要約生成を避けることができる

1. モデル構築

不具合報告表現言語モデルの再現は困難

- ソースコードや学習データ等が非公開・不足
- 実行ファイルのみが公開されており，編集不能

使用するモデル

学習済みBERTモデル (Bidirectional Encoder Representations from Transformers) * を用いる

- 自然言語処理の汎用的なモデル

>> 入力：不具合報告文の全文

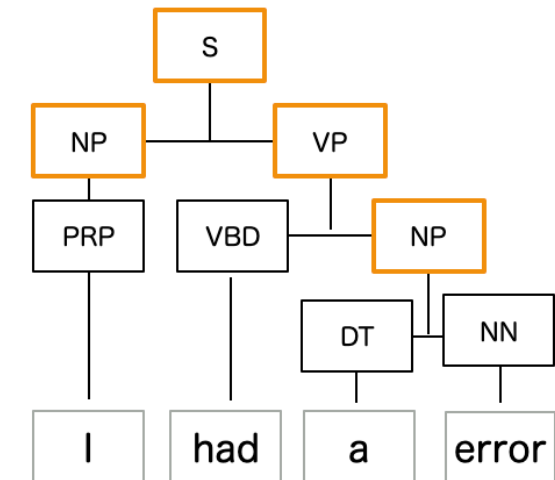
>> 出力：テキスト特徴量 (文章数 × 768次元 の高次元ベクトル)

* <https://github.com/hanxiao/bert-as-service#1-download-a-pre-trained-bert-model>

2. 句構造重みベクトル

1. 問題, 要因, 解決部分に該当する文を句構造に変換

- Githubで管理されている不具合報告から手動で収集 (200文)



2. 句構造タグを深さ優先探索で整列

- Type1: 葉ノードありRootあり
- Type2: 葉ノードありRootなし
- Type3: 葉ノードなしRootあり
- Type4: 葉ノードなしRootなし



※ 上記のPOSは構文タグを示す
ex) NP:名詞句, VP:動詞句, SBAR:従属節句

3. 整列させたデータで頻出するパターンを抽出 (句構造ルールと呼称)

2. 句構造重みベクトル

4. 抽出した句構造ルールを用いてベクトル化 (句構造重みベクトルと呼称)

不具合報告

句構造ルール i

$S_1 = [PP, S, SBAR, VP]$

$S_2 = [ADVP, NP, SBAR, VP]$ $Rule_i = [VP \rightarrow SBAR]$ $\rightarrow$ $Wscore_i = [0, 0, 1]$

$S_3 = [\underline{VP}, ADVP, \underline{SBAR}, NP]$

句構造重みベクトル $Wscore$

$$Wscore = \sum_{i=1}^n Wscore_i$$

n : 句構造ルールの数

$Wscore_i$: 不具合報告全文に句構造ルール i を
用いて重み付けを行なった句構造ベクトル

3. 重み付きテキスト特徴量

重みを付与したテキスト特徴量 WSF (Weighted Sentence Feature)

- WSF : 不具合報告文全文の特徴量
- $\tilde{WSF}$: 選択した文の特徴量

$$WSF = \sum_{i=1}^n S_i * Wscore_i$$

$$\tilde{WSF} = \sum_{i \in Chosen} S_i * Wscore_i$$

n : 不具合報告の文の数

S_i : 各文のテキスト特徴量

$Wscore_i$: 各文の重み

$Chosen$: 選択した文

WSF と $\tilde{WSF}$ の平均二乗誤差が最小になる組み合わせを選択
(ビームサーチ法)

評価実験1

重みを付与したテキスト特徴量 WSF を用いた場合,
要約文の抽出精度が向上するか

- 要約作成に重要であるというラベルが付与されている文の抽出精度

評価実験2

重みを付与したテキスト特徴量 WSF を用いた場合,
生成された要約文の理解しやすさが向上するか

- 問題解決過程を含む文の抽出精度

評価 | 方法

評価方法

重みを付与していないテキスト特徴量 SF を用いた場合と

重みを付与しているテキスト特徴量 WSF を用いた場合の要約抽出精度を比較

評価データセット

- データセットA [9] : 96件の不具合報告
- データセットB [10] : 36件の不具合報告
- 正解集合 (Gold Sentence Set) : 2人/3人 が要約作成に有用だと判断した文

[9] He Jiang, Jingxuan Zhang, Hongjing Ma, Najam Nazar, and Zhilei Ren. 2017. Mining authorship characteristics in bug repositories. Science China Information Sciences 60, 1 (2017), 012107.

[10] arah Rastkar, Gail C Murphy, and Gabriel Murray. 2014. Automatic summarization of bug reports. IEEE Transactions on Software Engineering 40, 4 (2014), 366–380.

評価指標

$$- Recall = \frac{|Chosen \cap GSS|}{|GSS|}, \quad Precision = \frac{|Chosen \cap GSS|}{|Chosen|}$$

GSS : 正解集合

$$- Fscore = \frac{|2 * Precision * Recall|}{Precision + Recall}$$

$Chosen$: 選択文

評価 | 重みありとなしの精度比較

	Recall	Precision	F値	重み正確性	内訳
データ セット A	重みなし(SF)	0.39	0.34	0.36	
	Type1	0.34	0.47	0.39	760/2500
	Type2	0.35	0.48	0.40	720/2400
	Type3	0.17	0.43	0.24	100/300
	Type4	0.13	0.43	0.20	40/120
データ セット B	重みなし(SF)	0.32	0.41	0.36	
	Type1	0.28	0.53	0.37	560/1400
	Type2	0.28	0.52	0.36	540/1300
	Type3	0.13	0.43	0.20	70/160
	Type4	0.08	0.47	0.14	25/70
	参考値 (Bugsum)	0.41	0.61	0.49	

Type1: Rootあり葉ノードあり, Type2: Rootなし葉ノードあり
 Type3: Rootあり葉ノードなし, Type4: Rootなし葉ノードなし

まとめと今後の予定

まとめ

- 問題解決部分を考慮した要約選択手法の提案
- 句構造重みベクトルを用いて文章に重みを付与することで精度の向上を目指す

今後の予定

- 追加要素
 - >>> Bertモデル → Bertsumモデルに置換
 - >>> ソースコード部分を要約選択の候補から削除する (Infozilla使用中)
 - >>> ROUGE (要約文の自然さ)
- なぜ精度が低下するのか, 変化があまりないのかを分析する
 - >>> 正解集合に含まれるルールと付与に使用したルールを比較する
- 評価実験2 (要約文に問題・要因・解決文が含まれているか)をする
 - >>> データ目視中

追加資料

Issue (課題管理票)

- GitHub(共同開発環境)に実装されている機能
- 開発製品の不具合や機能改善要求などの様々な議論内容を管理

CVE-2021-23840 still in recent Debian Buster based Python images #580 タイトル

Closed ahaerpfer opened this issue on 21 Feb · 2 comments

質問者 ahaerpfer commented on 21 Feb

Is it possible that the most recent builds for the Debian Buster based images were built from an outdated Debian package cache? Dockerfiles for the 3.8 and 3.9 versions were updated on Feb. 19 ([c285e28](#) , [91cbd74](#)), but the corresponding images on Dockerhub still contain the vulnerable `openssl 1.1.1d-0+deb10u4`.

On the other hand the [Debian Security Tracker](#) lists this vulnerability as fixed with the updated `openssl 1.1.1d-0+deb10u5` being available since Feb. 17 ([DSA-4855-1](#)), i.e. two days earlier!

When I build a `python:3.9-slim` image from the Dockerfile in this repo myself (today, i.e. Feb 20), I get the updated package.

See also [#578](#).

本文

回答者 ahaerpfer changed the title ~~CVE-2021-23840 still in recent Debian Buster images~~ CVE-2021-23840 still in recent Debian Buster based Python images on 21 Feb

wglambert commented on 23 Feb

Looks like the recent security update just missed it [docker-library/official-images#9658](#)

But there's another PR up [docker-library/official-images#9670](#)

コメント

yosifkit commented on 23 Feb

Unfortunately, [docker-library/official-images#9670](#) is not going to have an effect, since it only changes the pip installation. That happens after the installation of python and the builds make heavy use of Docker build cache, so the earlier layers will be unchanged when the images are built.

コメント

Labels
None yet

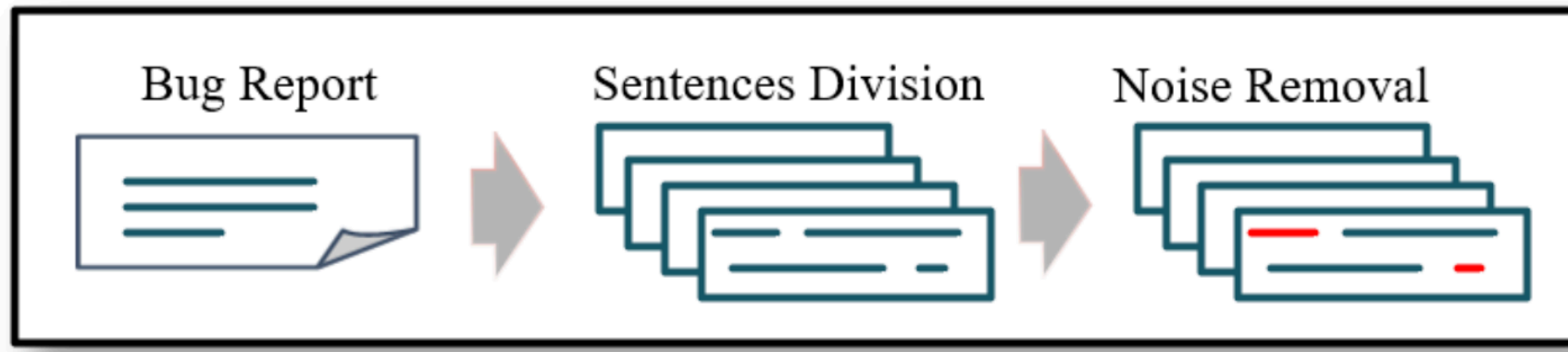
Projects
None yet

Milestone
No milestone

Linked pull requests
Successfully merging a pull request may close this issue.
None yet

4 participants

提案ツール | テキストデータの前処理



- 文字列の一部として句読点を使用している場合以外削除
 - ソフトウェア固有の正規表現 [2] (`([\w-]+(\.([\w-]+)*)`) を使用して文章をトークン化
 - 単語を正しく識別しながら関数名や変数名の大部分を保持する
- poster stemmer [3] を用いてステム処理
- ストップワード [4] 処理

[2] Rafael Lotufo, Zeeshan Malik, and Krzysztof Czarnecki. 2015.

Modelling the 'hurried' bug report reading process to summarize bug reports. Empirical Software Engineering 20, 2 (2015), 516-548.

[3] Martin F Porter. 1980. An algorithm for suffix stripping. Program 14, 3 (1980), 130-137.

[4] Damian Doyle. [n.d.]. Default English stopwords list. <https://www.ranks.nl/stopwords>. 2017.

先行研究モデルの再現

○ オートエンコーダの再現を行う

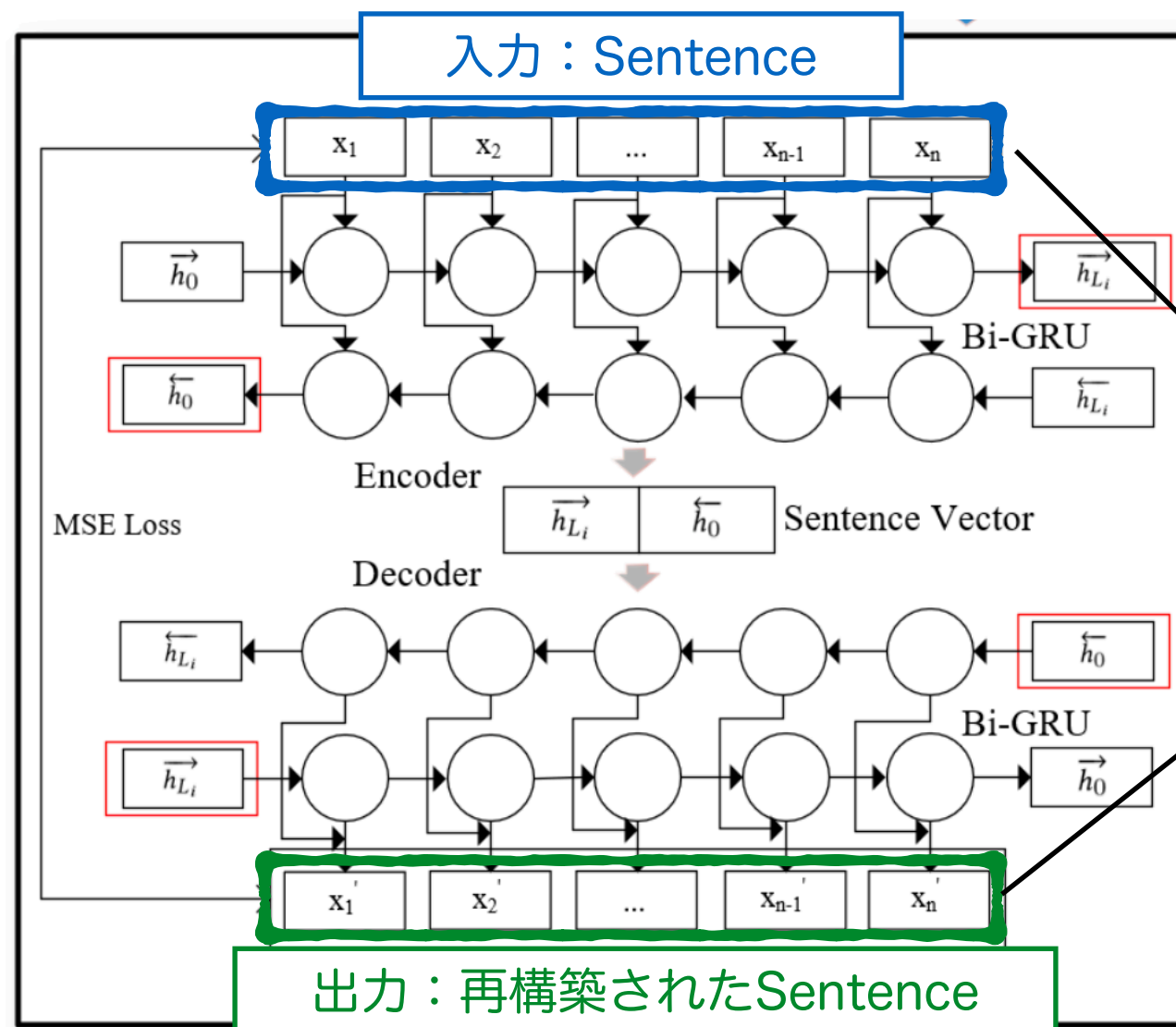
- 同一のプロジェクトの不具合報告データを学習させる
- フィルタを変えてもデータ数が一致ためノイズが少ないデータを使用

プロジェクト	先行研究	bug-Closed
HADOOP	6954	3835
ZOOKEEPER	1177	809
HIVE	2907	5363
DERBY	5705	3519
YARN	1826	871
HDFS	3710	1886
FLINK	8876	6735
計	31155	23018

単語の頻度に依存しない特徴量

○ オートエンコーダ (双方向GRU [51])

- 既存手法は単語の頻度に依存することにより, 要約文が冗長
- オートエンコーダを用いて, 単語に依存しないテキスト特徴量を抽出



x_t : 1文に含まれる1単語

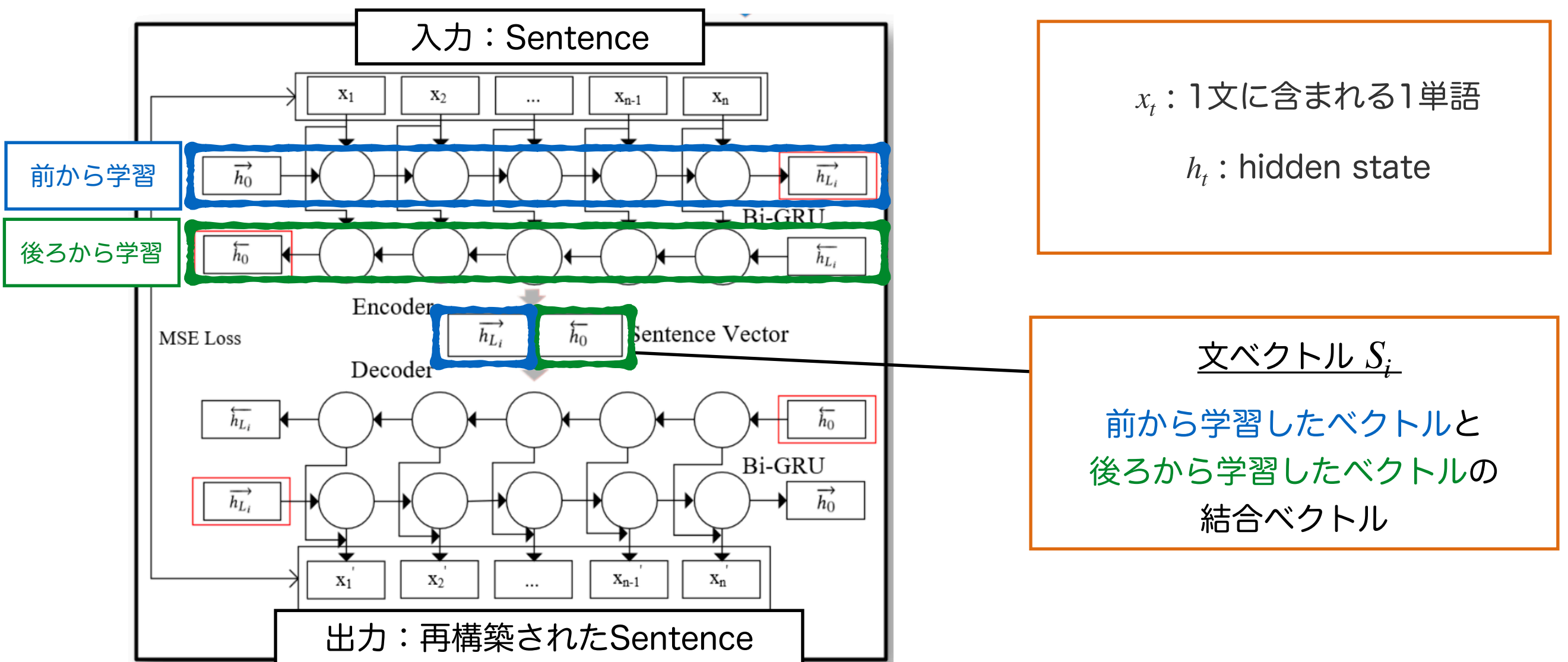
h_t : hidden state

入力文と出力文の平均二乗誤差が
最小になるように学習
(学習率 : 0.01)

単語の頻度に依存しない特徴量

○ オートエンコーダ (双方向GRU [51])

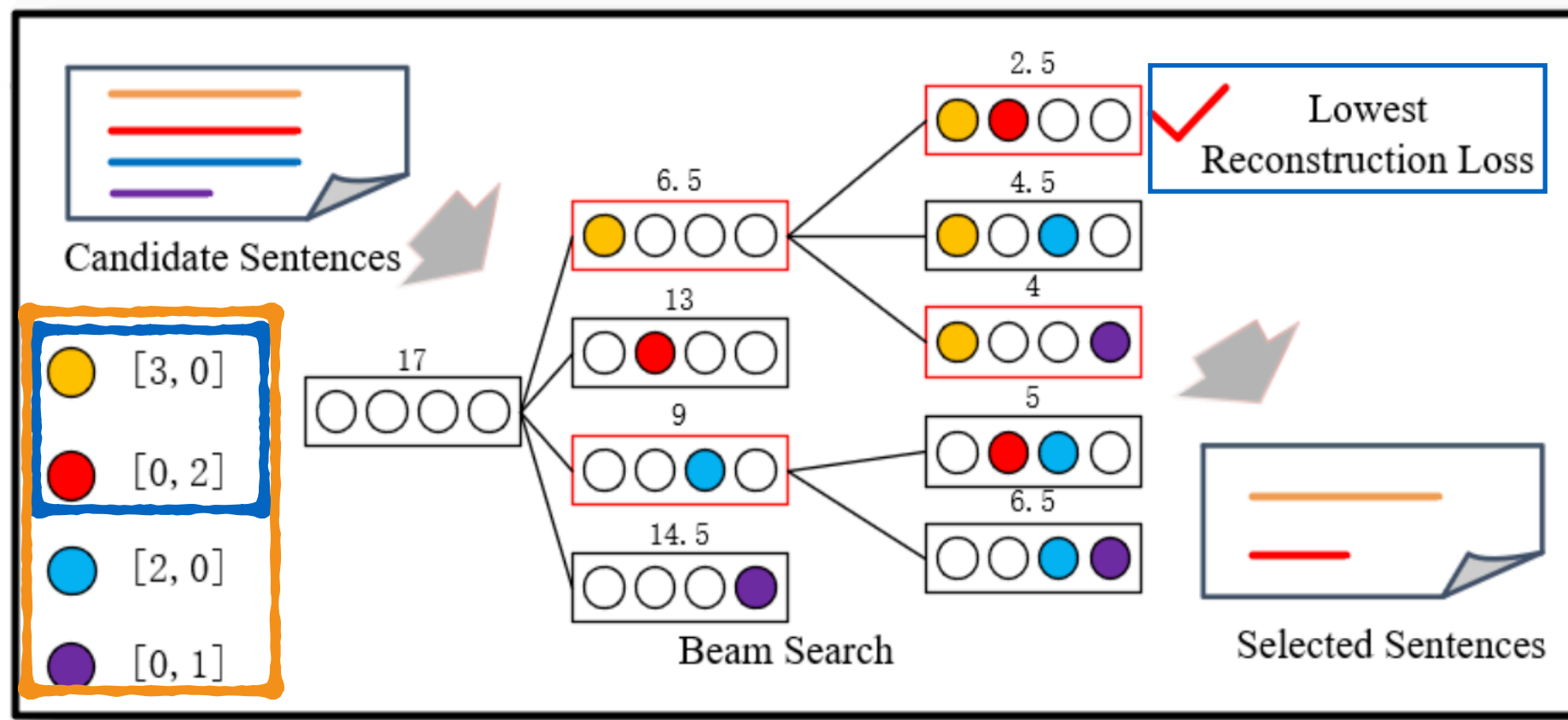
- 既存手法は単語の頻度に依存することにより, 要約文が冗長
- オートエンコーダを用いて, 単語に依存しないテキスト特徴量を抽出



要約選択

○ 以下の2つ平均二乗誤差が最小になる組み合わせを算出する

- 1つの不具合報告文全ての特徴量 DF
- 選択した文の特徴量 $\tilde{DF}$



$$DF = [5.3]$$

$$DF_1 = [3, 2]$$

$$\delta_1 = \frac{(5 - 3)^2 + (3 - 2)^2}{2} = 2.5$$

○ 例では1文を2次元特徴量で表現

○ ビームサイズは8で行う

構文タグセット

NP	名詞句	SQ	Yes/noの質問とwh要素を除くSBARQの従属句
PP	前置詞句	WHASVP	副詞のフレーズ
ADVP	副詞句	WHNP	Wh名詞句
ADJP	形容詞句	WHPP	Wh前置詞フレーズ
VP	動詞句	X	不明
S	宣言文句	0	従属節のthatの変化形
SBAR	従属句	T	Wh構成要素
SBARQ	直接質問	“	不定詞や命令形の理解された主語
SINV	主語と副詞が逆転している宣言句		

* <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.8216&rep=rep1&type=pdf>

品詞タグセット

CC	等位接続詞	JJR	形容詞 (比較級)
TO	不定詞のto	WDT	Wh限定詞
CD	基数	LS	List item marker
UH	介詞	WP	Wh 名詞句
DT	決定詞	MD	モーダル
VB	動詞 (原形)	WP\$	所有格のある代名詞
EX	存在のthere	NN	名詞 (単数)
VBD	動詞 (過去形)	WRB	Wh 形容詞句
FW	外来語	NNS	名詞(複数)
VBG	動詞 (動名詞, 現在形)	NNP	固有名詞
JJ	形容詞	PRP	人称代名詞
VBP	動詞 (3単元)	RBS	副詞 (最上級)

など

* <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.8216&rep=rep1&type=pdf>

前処理

○ 前処理見直し

- 前処理からルール抽出, 要約までの流れを見直し(リファクタリング)
- Lemma化
- URL除去
- Ignored適用 (Bugsum参考. 増やす予定)
- stopword除去 (spacy [en_core_web_sm], Bugsum適用単語)
- Token化 (空白 or 正規表現)

○ Ignoredで要約選択の際に無視する文

- Infozillaでヒット (ソースコード除去がこれだけなので増やす)
- Stopword除去後に英文字を含まない
- 先頭に > or >を含み類似度90%の文章
- 単語数が50以上 or 英文字10語でTimeStampを含む

ルール数

葉ノードを含む

- Lift 1以上, Support **0.80以上**
- 灰色: 要素が出現するルール数
- 黒色: 固有ルール数

Lift : 関連性

Support : 出現割合

Root	問題	要因	解決	その他
あり	60	1500	600	4
あり	0	1400	150	-
なし	60	1500	600	4
なし	0	1000	40	-

* 不明POS 「X」を含むルールも含む

ルール数

葉ノードを含まない

- Lift 1以上, Support **0.06以上**
- 灰色: 要素が出現するルール数
- 黒色: 固有ルール数

Lift : 関連性

Support : 出現割合

Root	問題	要因	解決	その他
あり	900	2400	1374	1324
あり	78	1538	492	-
なし	600	1100	1300	376
なし	0	408	556	-

* 不明POS 「X」を含むルールも含む

具体例

