

文章的要素を考慮した不具合報告の 要約生成ツールの提案

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
 - バグリポジトリ (Bugzilla), 不具合追跡システム (github) 等
- 効率的に不具合に関する情報を開発者は把握する必要がある
 - 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 [42]
 - 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 [42]
- 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

背景 | 要約方法

○ 要約方法は2種類存在する

○ 抽出型

- 要約対象の文章の中から重要と思われる文を抽出して要約を作成する手法
- 特徴
 - 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

○ 抽象型

- 文章の意味を汲み取ったうえで(=抽象化した上で)適切な要約を作成する手法
- 特徴
 - 文中にない単語を利用できるため自由度の高い要約を作成できる
 - 自然な文章を生成することが困難

背景 | 抽出型要約の関連研究

○ バグ報告の要約は教師あり or 教師なしの機械学習で実装

- 時間的コストを削減するためにバグ報告を要約する研究が行われている

○ 教師あり [17, 43]

- 学習コーパスの品質に依存
- 膨大な手作業
- 一定の専門知識を持つアノテーターが必要

○ 教師なし [22, 30, 52]

- 単語の頻度に過度に依存
- 同じ立場を支持するコメント群は類似しているため冗長な要約文が作成される

先行研究 | 既存手法とその課題点

○ BugSum (先行研究の要約手法)

- 要約の冗長性を軽減し、信頼性のある要約を生成する要約手法を提案 [2]
- 従来手法：冗長 + コメントの信憑性を考慮していない
- 単語の頻度に依存せず、コメントの信憑性を加味して要約文を選択

○ Bugsumは、不具合報告内の問題点や要因、解決策を考慮していない

- 不具合報告を正確に要約できていない可能性がある
- 各要素 (問題, 要因, 解決) の構造的特徴を用いて要約文を選択する

着眼点

構造的特徴を句構造パスとして表現し
各要素を含む要約文を選択する

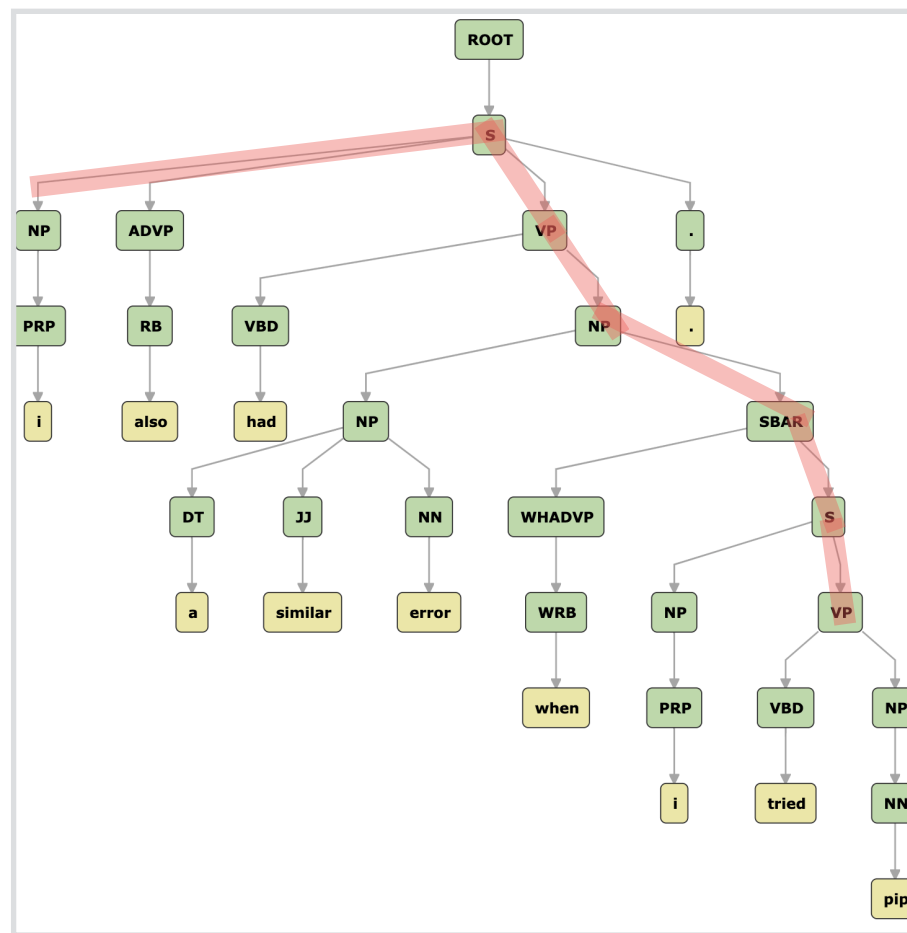
[2] Haoran Liu, Yue Yu, Shanshan Li, Yong Guo, Deze Wang, and Xiaoguang Mao. 2020.
BugSum: Deep Context Understanding for Bug Report Summarization.
In Proceedings of the 28th International Conference on Program Comprehension (ICPC '20).
Association for Computing Machinery, New York, NY, USA, 94–105. DOI:<https://doi.org/10.1145/3387904.3389272>

着眼点 | 句構造パス

- 各要素 (問題, 要因, 解決) の文構造は異なるという仮説
→ 文構造を句構造パスとして表現する

1文

句構造



I also had a similar error
when i tried pip.

最も経路が長いもの
とりあえず1つの文章から1つ
各単語の品詞タグではなく、構文タグのみ
(葉とその一つ上の単語のタグを除いて)

句構造パス



※ 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

本研究の目的

目的

不具合報告の概要を正確に効率よく
開発者が理解すること

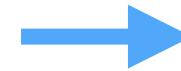
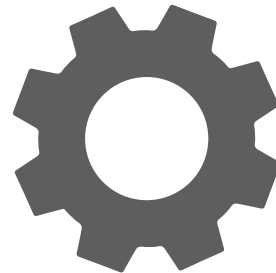
→ 不具合報告の要約生成ツールの提案
(元の文章から重要な文章を抜き出す抽出型)



不具合報告

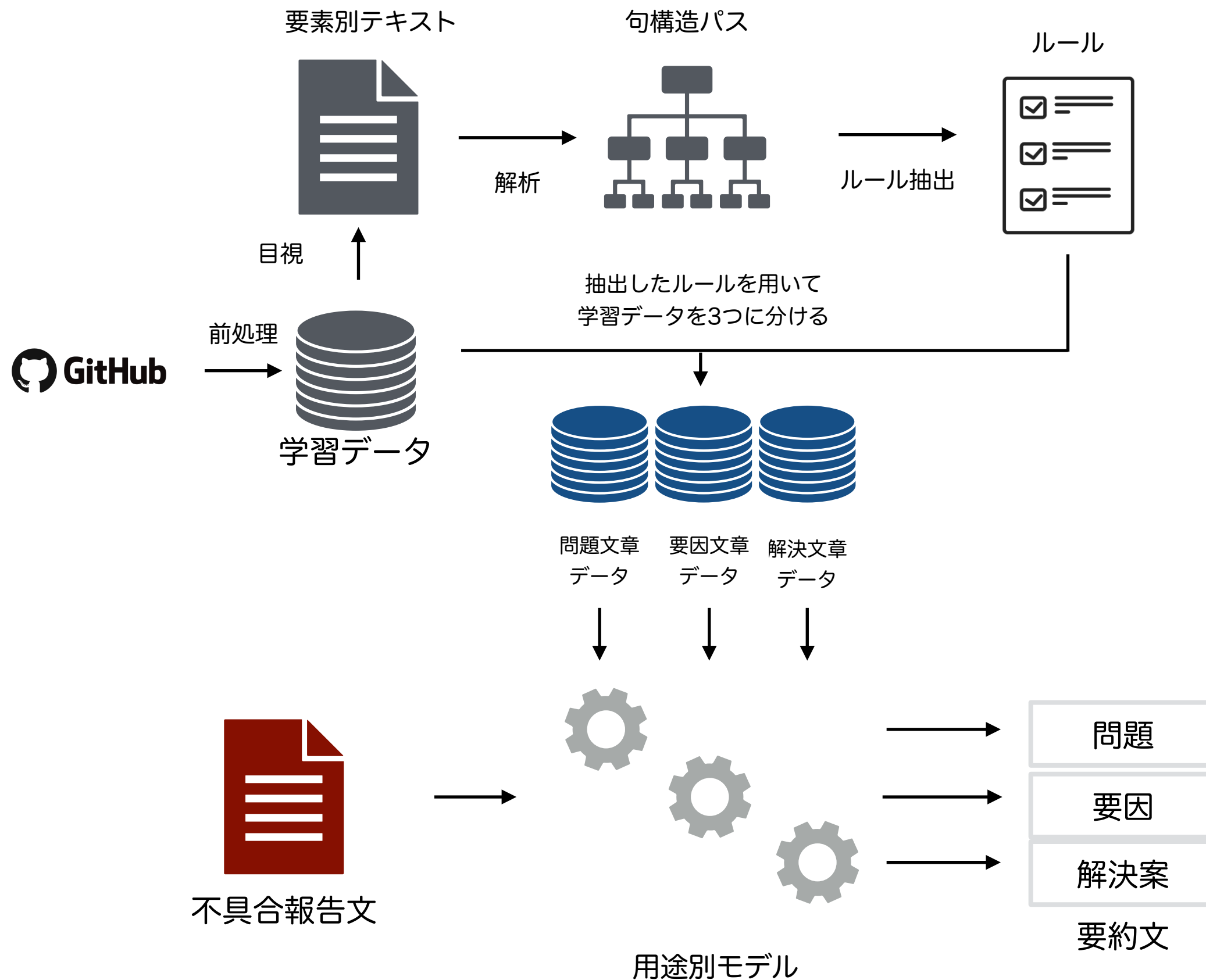


提案ツール



問題, 解決に関する文章を
抽出する

提案ツールの概要



提案ツールの概要

○ 単語の頻度に依存しないテキストの特徴量

- 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

○ 要素別モデルを作成

- 既存手法は文の要素を考慮していない
- 問題，要因，解決策に関する文を抽出することに特化したモデルを作成
- 句構造パスに対してルールマイニングを行い，学習データを分類する

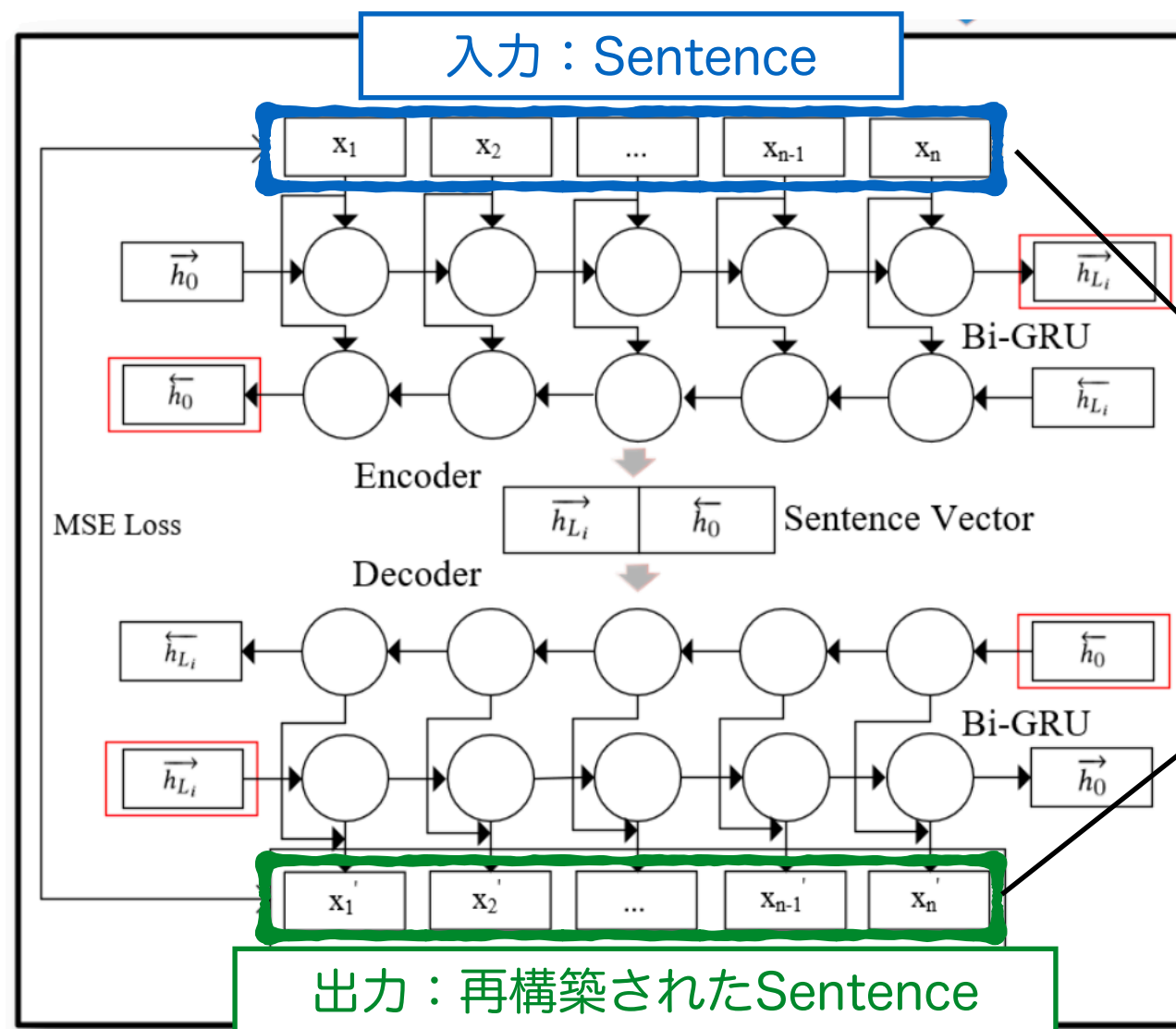
○ 要約文の選択

- ビームサーチ法を用いて要約文を選択する

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

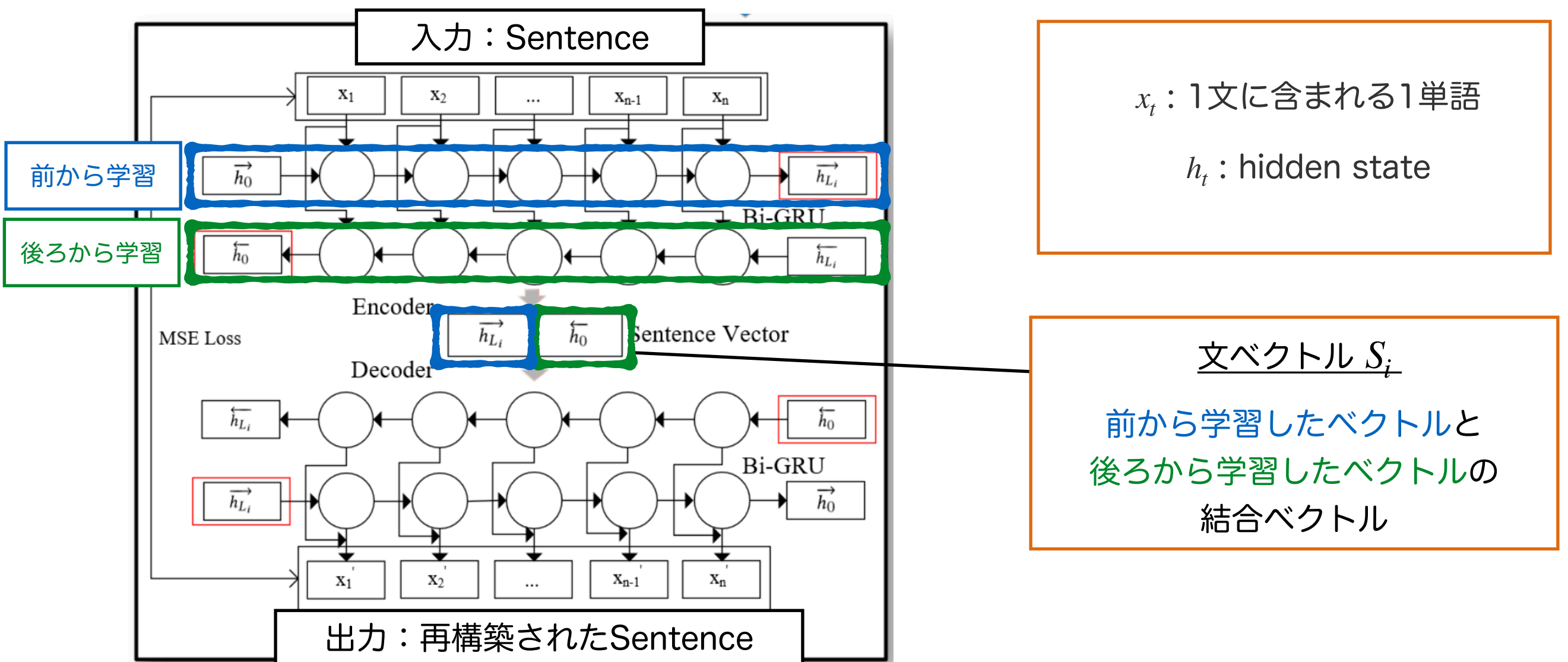
h_t : hidden state

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

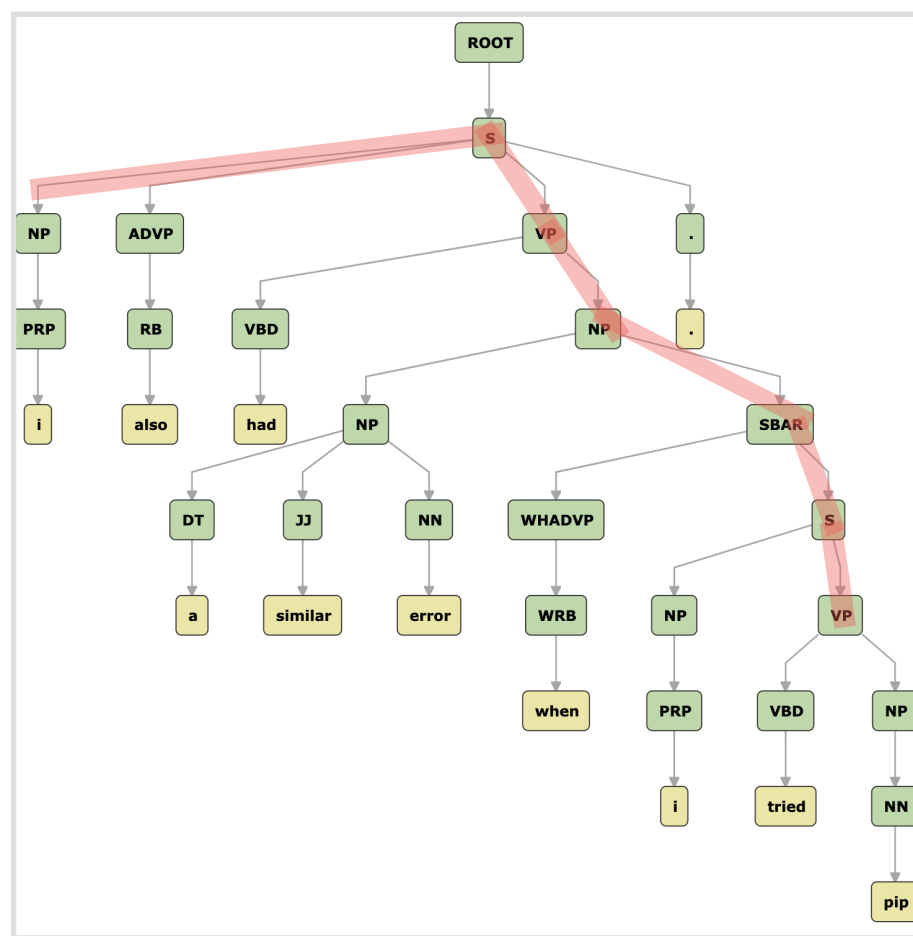


要素別モデル | 学習データの分類 (再掲)

- 各要素 (問題, 要因, 解決) の文構造は異なるという仮説
→ 文構造を句構造パスとして表現する

1文

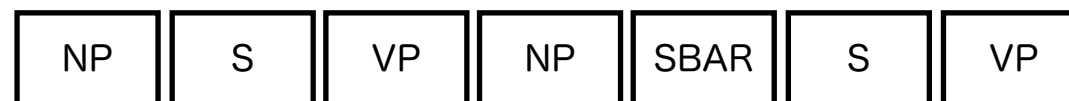
句構造



I also had a similar error
when i tried pip.

最も経路が長いもの
とりあえず1つの文章から1つ
各単語の品詞タグではなく、構文タグのみ
(葉とその一つ上の単語のタグを除いて)

句構造パス



※ 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

要素別モデル | 学習データの分類

○ github issueを目視して以下の4つにデータを分割

- 問題, 要因, 解決, その他

○ 目視データから句構造パスのルールを抽出する

→ Lift 1以上, Support 0.06以上

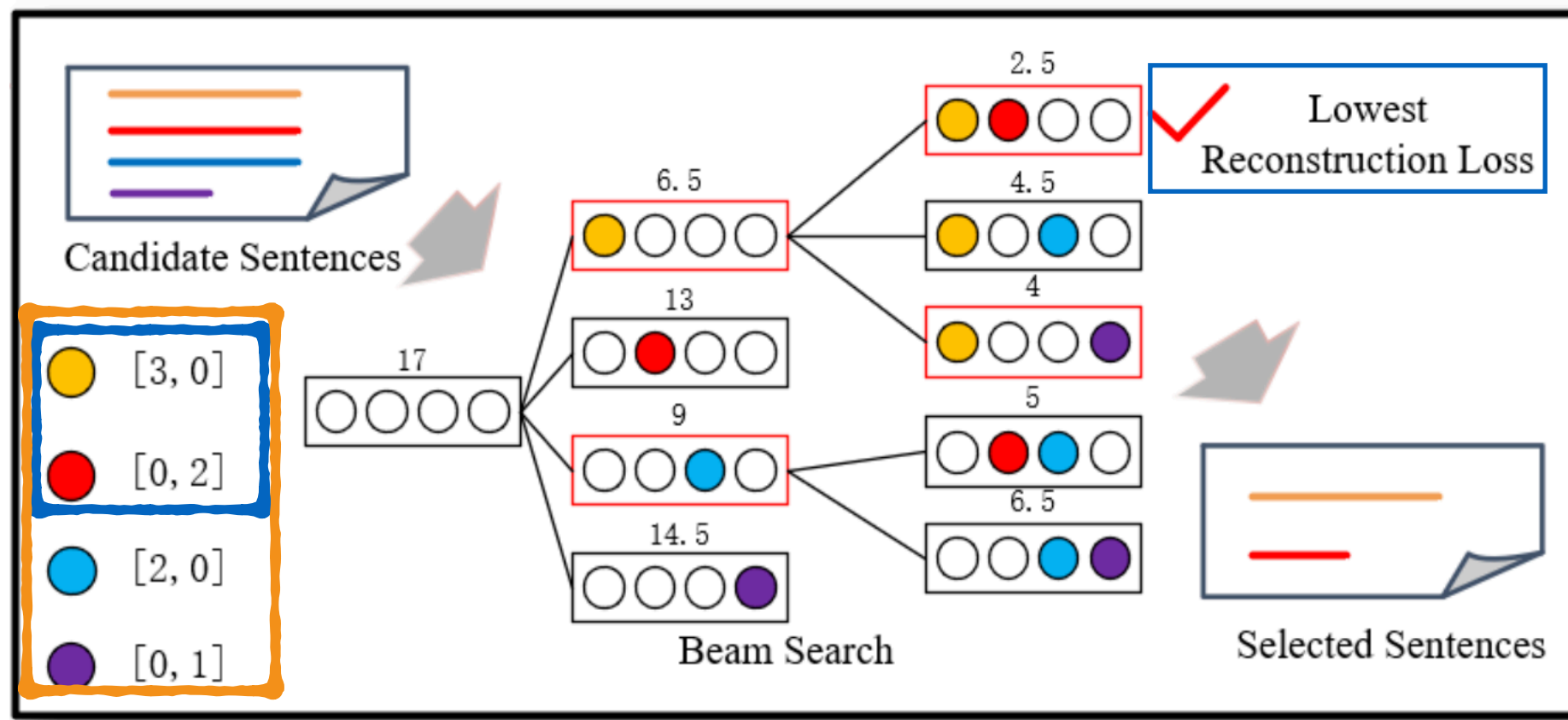
問題	48
原因	342
解決策	156
その他	58

抽出したルールを用いて学習データを分類し,
用途別のモデルを構築する

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つのバグ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

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

先行研究モデルの再現

○ 2種類のベンチマークデータセットを用いて学習したモデルの評価

- 36件のバグ報告：Summary Dataset (SDS) [43]
- 96件のバグ報告：Authorship Dataset (ADS) [17]
- 3人のアノテータが目視を行う
- 2人/3人が要約作成に有用だと判断した文をGold Sentence Setとする

○ 精度

- $Recall = \frac{|Chosen \cap GSS|}{|GSS|}$: GSSが要約にどの程度使われているか

- $Precision = \frac{|Chosen \cap GSS|}{|Chosen|}$: 選択された文の中にどの程度GSSが含まれているか

先行研究モデルの再現 | 結果

○ 先行研究との違い

- 信頼性に関することはしていない

	Recall	Precision
先行研究	0.41	0.62
再現 (文単位)	0.22	0.39
再現 (単語単位)	0.35	0.55

$$Recall = \frac{|Chosen \cap GSS|}{|GSS|} : \text{GSSが要約にどの程度使われているか}$$

$$Precision = \frac{|Chosen \cap GSS|}{|Chosen|} : \text{選択された文の中にどの程度GSSが含まれているか}$$

○ モデルの学習&読み込みが正確にできていなかった

○ 先行研究のテキストの信頼性に関する要素がまだ導入できていない

現在していること

○ モデルの再学習 & 読み込みを正確に

- パラメータが先行研究と同じぐらいに設定できていなかった
- テキストの特徴量を表現できてなかった？ → 精度に影響があったのでは

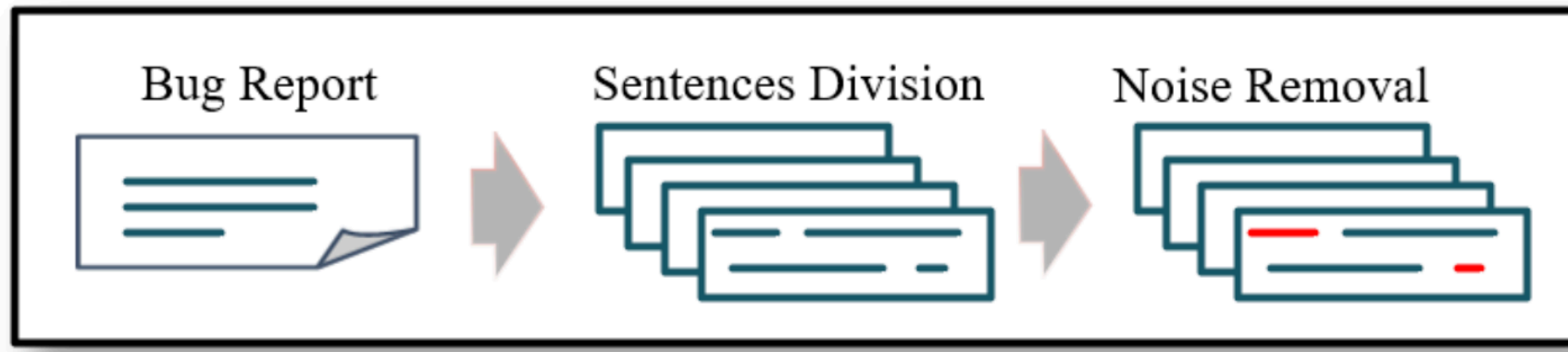
○ 信頼性に関する要素を取り込む

- 再現学習データの中に否定 or 肯定を分類するSVMを作成するのに
使えそうな学習データがあるので試してみる

○ 再現結果が先行研究とある程度近くなれば用途別モデルを 構築して精度比較したい

追加資料

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.

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

質問者

本文

コメント

コメント

回答者