

バグ報告の要約文の作成

BugSum: Deep Context Understanding for Bug Report Summarization

Haoran Liu, Yue Yu , Shanshan Li, Yong Guo, Deze Wang, Xiaoguang Mao
ICPC '20, October 5–6, 2020, Seoul, Republic of Korea

背景

○ バグ報告内の議論は多く，効率的に必要な情報を知りたい

○ 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

- 教師あり：

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

- **教師なし**：単語の頻度に過度に依存（一部のコメントは削除される，
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）

○ 新たな教師なしバグ報告要約法の提案

- 文をベクトルに変換し意味的特徴を抽出

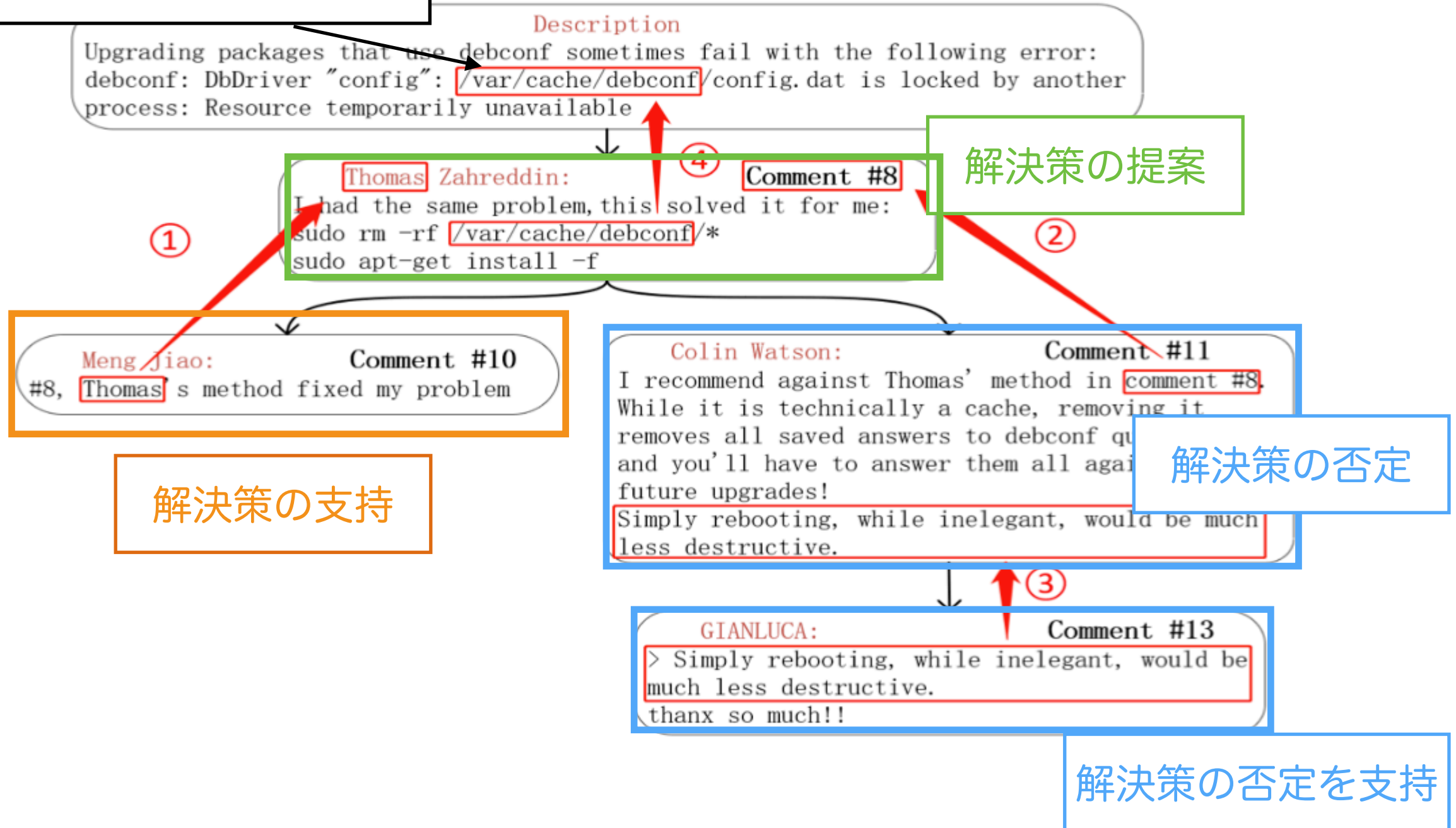
- 文章が議論の中で承認されたかどうかの度合いを図る指標を提案

→ 議論の余地のある文章が要約に選ばれる可能性を低減

- 高品質な要約を生成する候補文には以下の二つが大事
 - 含む情報 (informativeness)
 - 信憑性 (believability)
- 既存手法は単語の頻度に過度に依存している
 - 同じ立場を支持するコメント群 = 既存手法で要約に使われる確率が高い [13]
 - 同じ意味を持つ異なる文章を選択する → 冗長な要約文になってしまう
- 信憑性のないコメントは要約生成に適切でない
 - 信憑性のないコメントはバグ報告に頻出する
 - 単語頻出アプローチをする従来手法では適切な要約が作成されない

例 | 信憑性 (believability)

ドメイン単語：コメントの紐付け

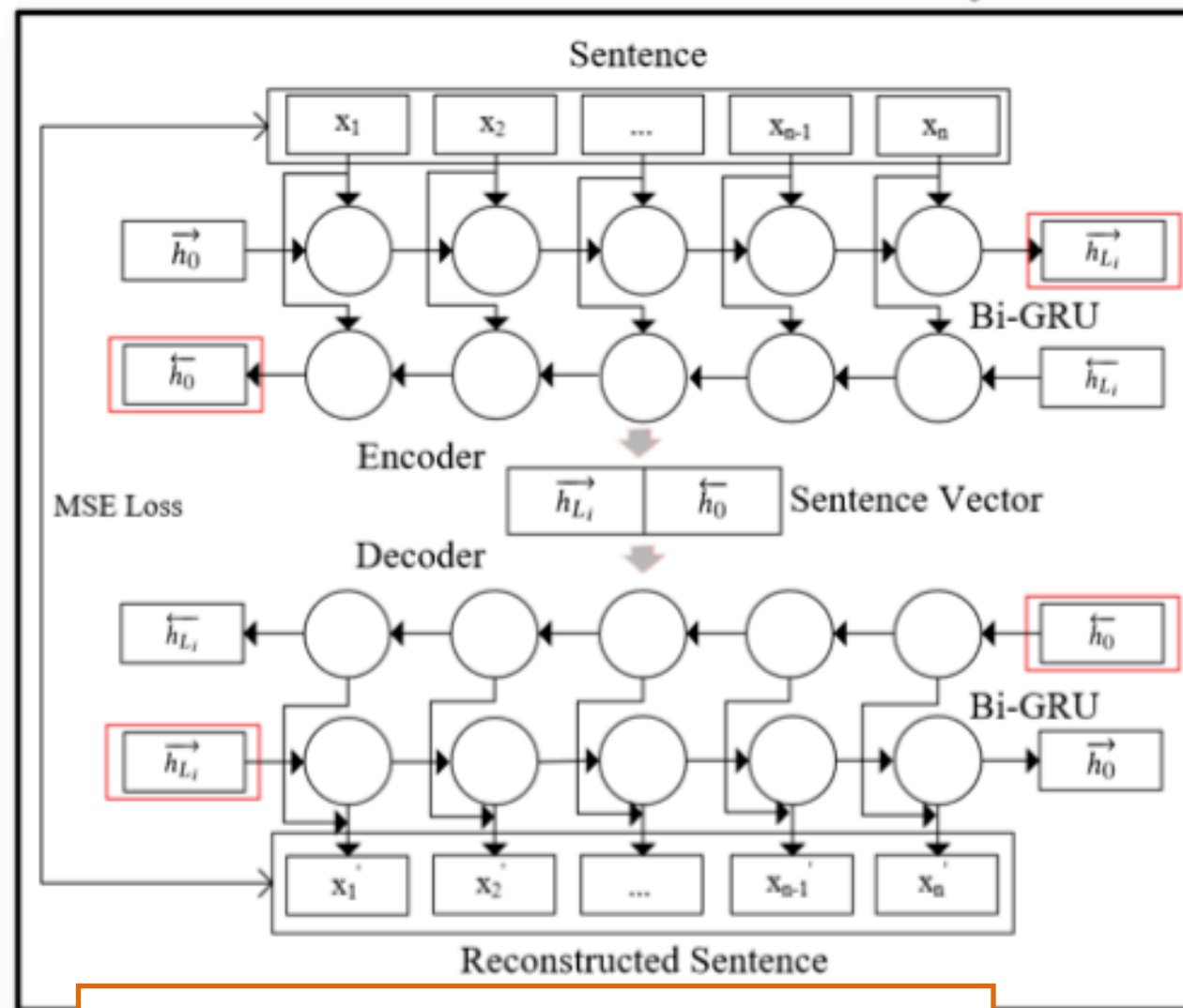
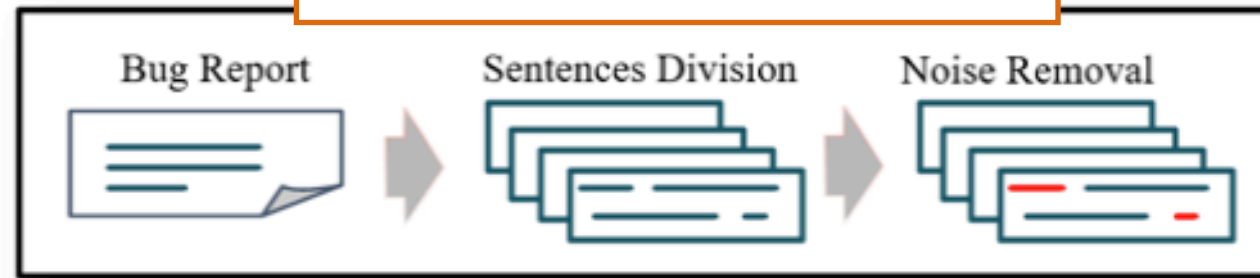


物議を醸すコメント：#8

- 報告文に頻出する → 過去の単語頻出ベースのアプローチでは要約に偏りが生じる

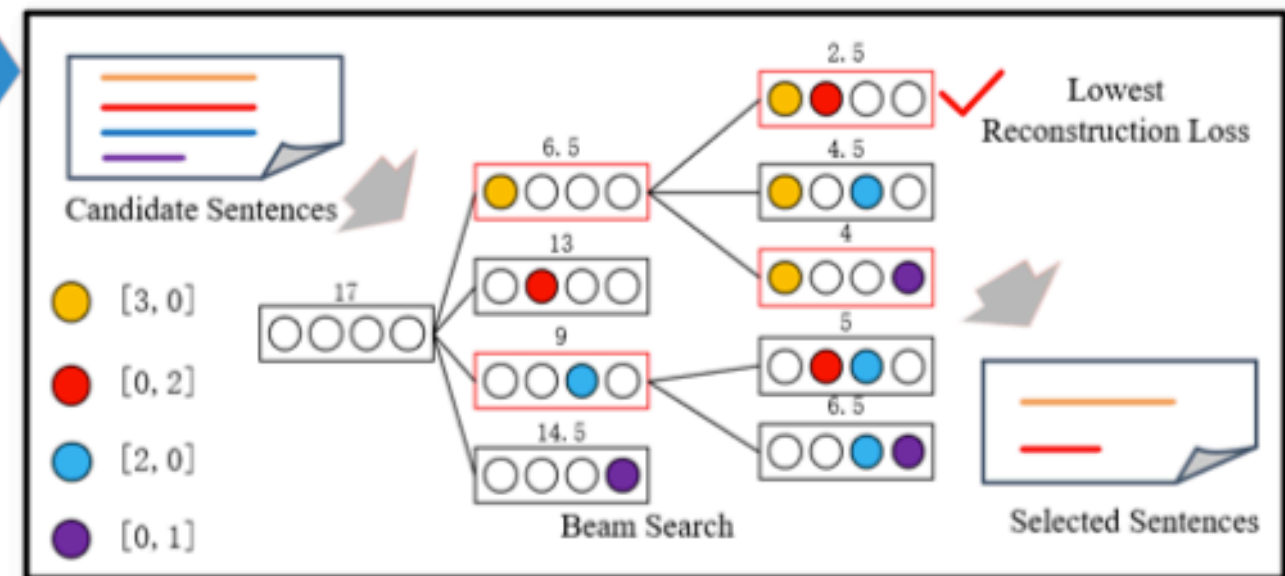
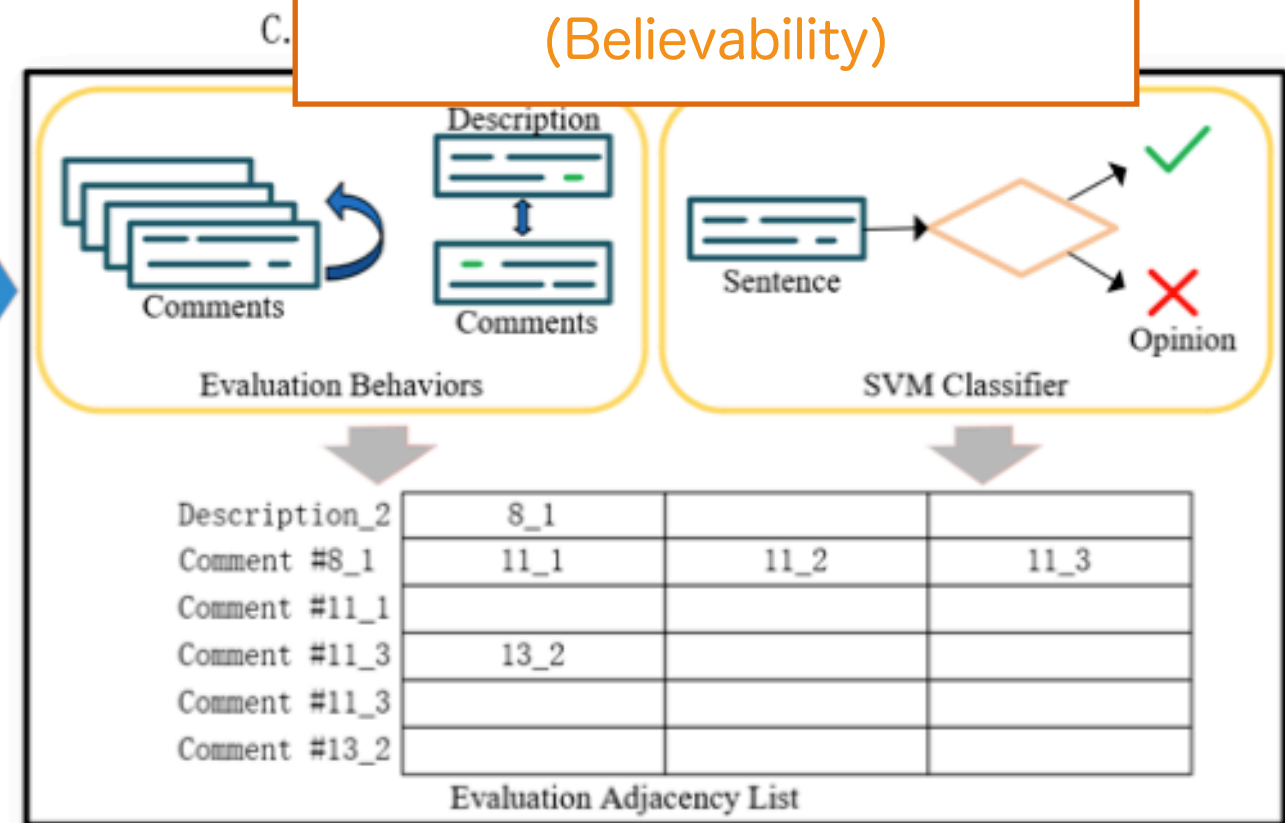
提案手法の概要図

バグ報告の前処理



オートエンコーダを用いて
文のテキスト特徴量抽出 (Infomativity)

各文章の信頼性をスコアリング (Believability)



要約作成のための文選択

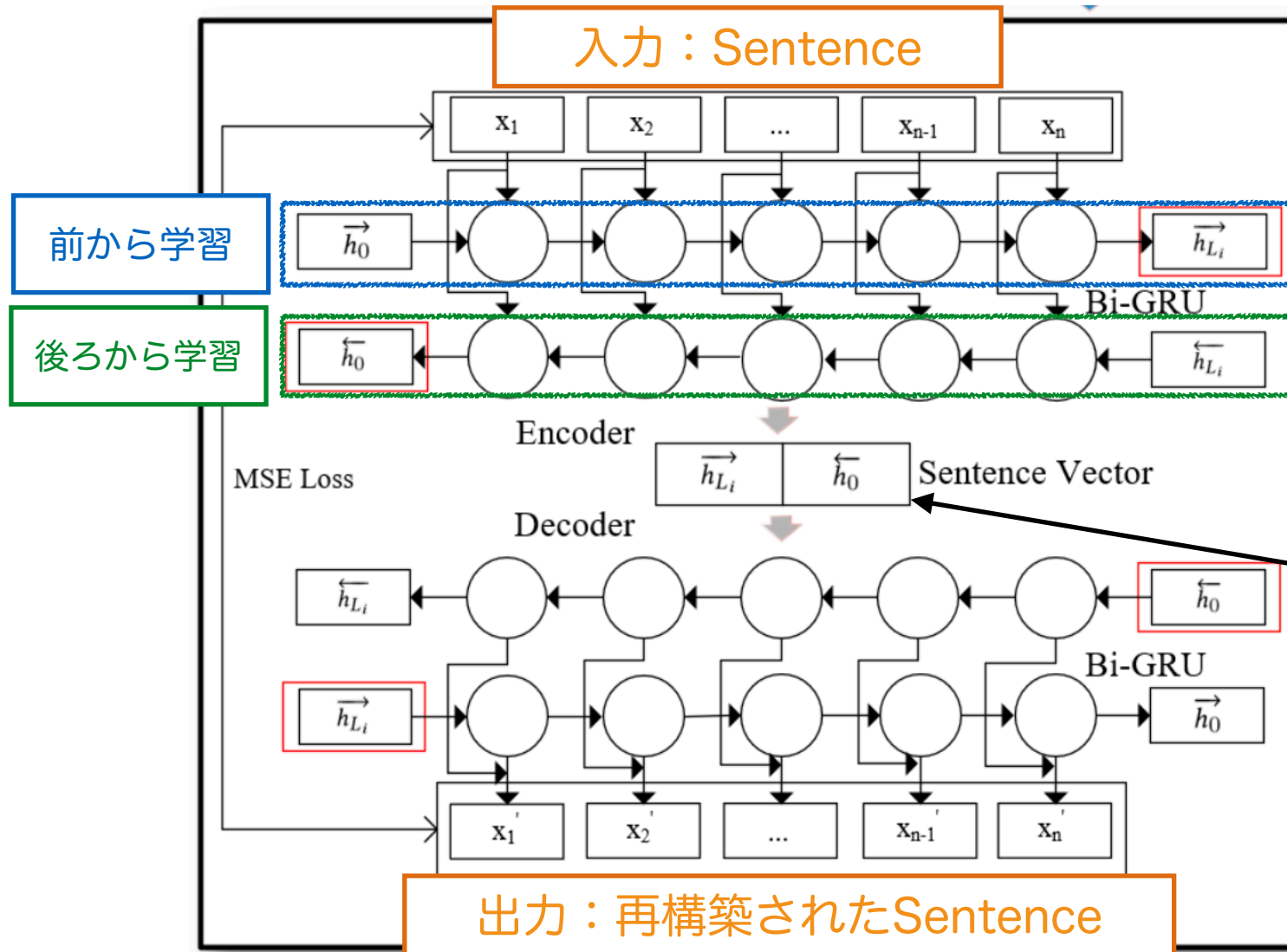
提案手法 | 前処理

- 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^[27]を使用して文章をトークン化
 - $[\backslash w-]+(\backslash.[\backslash w-]+)^*$
 - 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
- porter stemmer^[39]を用いてステム処理
- ストップワード^[10]処理

提案手法 | Informativeness

○ オートエンコーダ

- リカレントユニットには双方向GRU (Bi-GRU) [51]を採用 ← LSTMの拡張版



○ x_t : 1文に含まれる1単語

○ h_t : hidden state

文ベクトル S_i :

前から学習したベクトルと
後ろから学習したベクトルの
結合ベクトル

MSEが最小になる S_i = 文章の特徴を正確に表現している
(学習率0.01に設定)

提案手法 | Believability

○ 文 s_i のBelievability Score

- $EAdj_i > 0$: $Bscore_i = 1 + \sum_{j \in EAdj_i} Bscore_j * OPscore_j$
- $EAdj_i = 0$: $Bscore_i = 1$
- また $Bscore_i \geq 0$

$EAdj_i$: 評価コメントの数

$OPscore_j$: 否定的-1 or 肯定的1

○ 事前に学習されたSVM分類器でコメントの支持不支持を判断

- 支持不支持の文それぞれ1500文(手でラベル付け)を学習
- 直接的な返答はできないためドメイン単語があるかで判断

○ 評価コメントは明示的なコメントのみ対象

提案手法 | スコアリング (Believability + informativeness)

○ 文 s_i のスコアリング

- 情報量 + 信憑性
- バグ報告内の文章すべての特徴量(テキスト×信憑性スコア) : the domain textual features

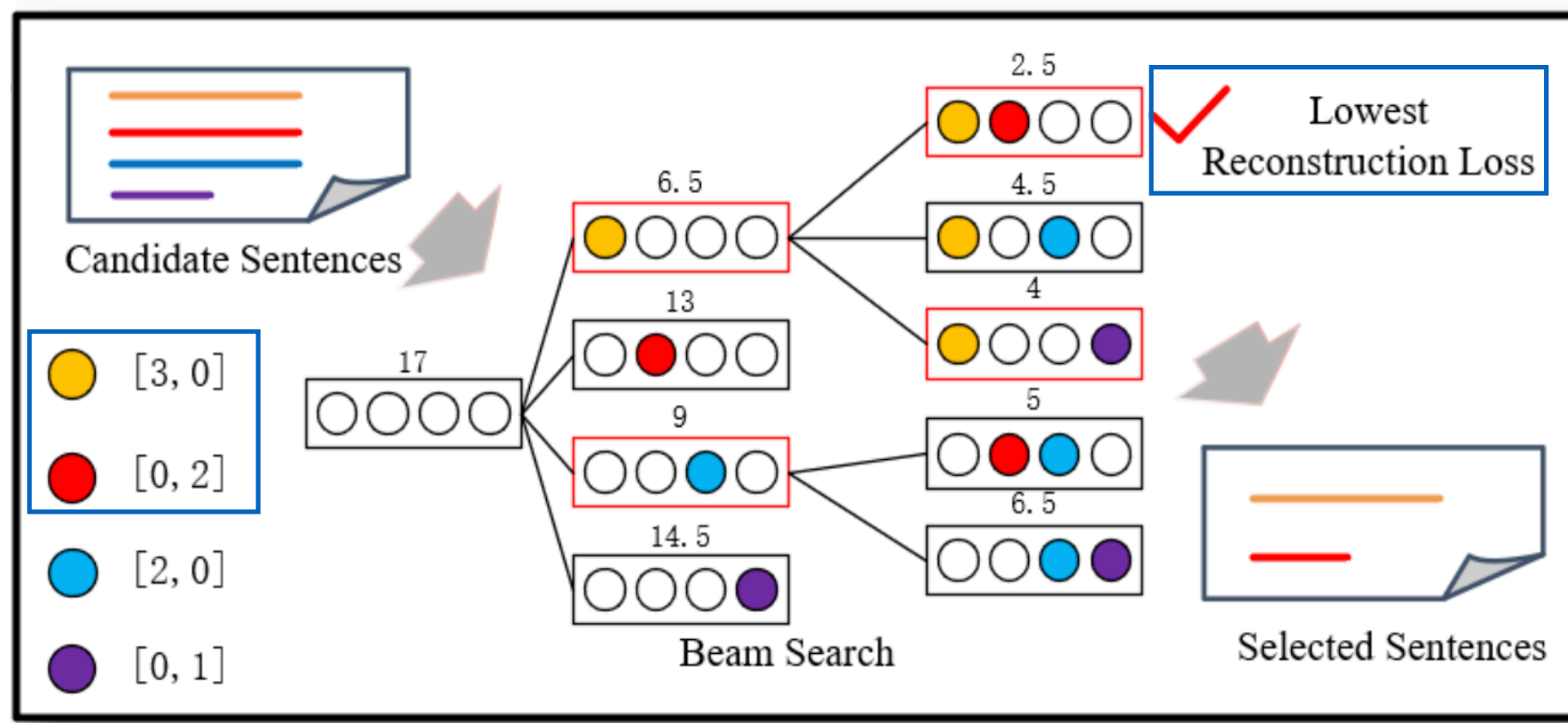
$$DF = \sum_{i=1}^n Bscore_i * S_i$$

n : バグ報告内の文の数

S_i : 文ベクトル

提案手法 | 要約の作成

- バグ報告の文全ての特徴量 DF と選択した文の特徴量 $\tilde{DF}$ の平均二乗誤差が最小になる組み合わせを算出する



$$DF = [5.3]$$

$$DF_1 = [3, 2]$$

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

- 文章が n 個あるバグ報告では 2^n 通りの要約が生成される = NP困難問題
→ 近似アプローチをしてビームサーチアルゴリズムを用いる

○ RQ1

- BugSumはベースラインのアプローチに対してどのように機能するか

○ RQ2

- BugSumは要約に選ばれる物議を醸すコメントを減らすことができるか

○ RQ3

- パラメータはBugSumのパフォーマンスにどのように影響するか

○ RQ4

- 文の特徴抽出と動的選択はBugSumの性能にどのような影響を与えるか

評価 | データセット

○ 2種類のベンチマークデータセットを用いて提案手法を評価

- 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

○ データセットの内容

- バグ報告の25%程度の抽象的な要約 (AbsSum)
 - 要約作成に最も役に立った文
- 2人以上がリストした文をGSS (the golden standard sentences set)[43] とする

評価 | 指標

○ 精度

- $Recall = \frac{|Chosen \cap GSS|}{|GSS|}$: GSSがどの程度選択されているか

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

- $Pyramid = \frac{Num_{ChosenListed}}{Num_{MaxListed}}$ [36]

- 複数人のアノテータがいる場合に要約の品質をよりよく測定する
- より多くのアノテータによってピックアップされたセンテンスが優先されているか

○ 読みやすさ

- Rouge-1, Rouge-2 [23]

- 作成した要約とAbsSumの間に連続的に重複するユニット数で品質評価

評価 | 比較

ウィルコクソンの符号順位検定 [15]より
p値<0.05でBugSumがベースラインアプローチを上回っていることを示す

Table 1: Overall Performance on SDS

SDS	F-score	Precision	Recall	Pyramid	R-1	R-2
Centroid	0.343	0.536	0.270	0.460	0.472	0.126
Grasshopper	0.369	0.527	0.301	0.523	0.509	0.135
DivRank	0.378	0.590	0.301	0.545	0.527	0.138
ACS	0.397	0.596	0.335	0.600	0.515	0.134
BRC	0.401	0.572	0.351	0.629	0.522	0.140
Hurried	0.410	0.711	0.300	0.710	0.527	0.153
MMR	0.429	0.617	0.353	0.551	0.498	0.145
DeepSum	0.462	0.621	0.388	0.624	0.563	0.177
BugSum	0.493	0.629	0.413	0.661	0.589	0.194

Table 2: Overall Performance on ADS.

ADS	F-score	Precision	Recall	Pyramid	R-1	R-2
DivRank	0.325	0.446	0.282	0.542	0.499	0.201
Centroid	0.337	0.488	0.280	0.561	0.473	0.183
MMR	0.396	0.505	0.356	0.585	0.503	0.206
Grasshopper	0.361	0.445	0.337	0.546	0.503	0.200
BRC	0.411	0.566	0.349	0.656	0.516	0.206
Hurried	0.417	0.576	0.346	0.635	0.540	0.239
ACS	0.453	0.609	0.396	0.672	0.546	0.231
DeepSum	0.457	0.606	0.394	0.681	0.553	0.249
BugSum	0.491	0.611	0.417	0.692	0.564	0.270

頻出単語の依存した手法とは異なり
選択した文の網羅性を考慮して
文を選択している

→ ノイズ文の選択を一部防いだ

意味的な冗長性を減らすことで、
より多くの重要な文章をカバーできる

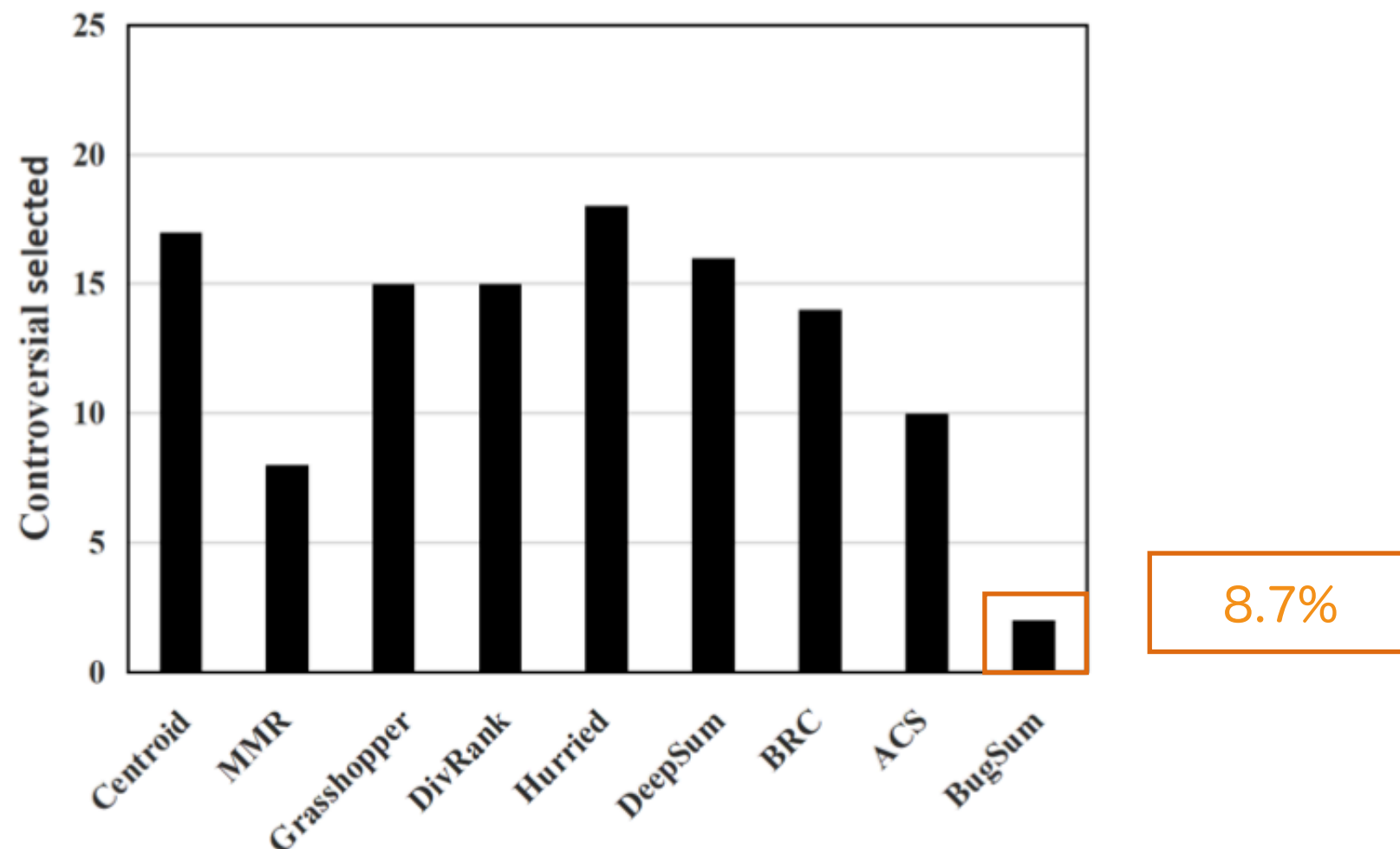
所感

- 実際どう言う文章からどういう要約ができたみたいな例がなかったので
実例を見せて欲しかった
- 自分の研究もissueの要約作成へシフトしたので非常に参考になる研究だった
- 論文の流れとしては綺麗だなと思った

RQ2 以降

RQ2 | 物議を醸すコメントを減らせるのか

- SDSとADSからそれぞれ7個と16個の物議を醸すコメントを目視
 - プログラム経験4年以上の5人
 - その文が少なくとも1つ以上のベースラインアプローチで選択されている
 - その文の評価分の全てが否定的なコメントであるか

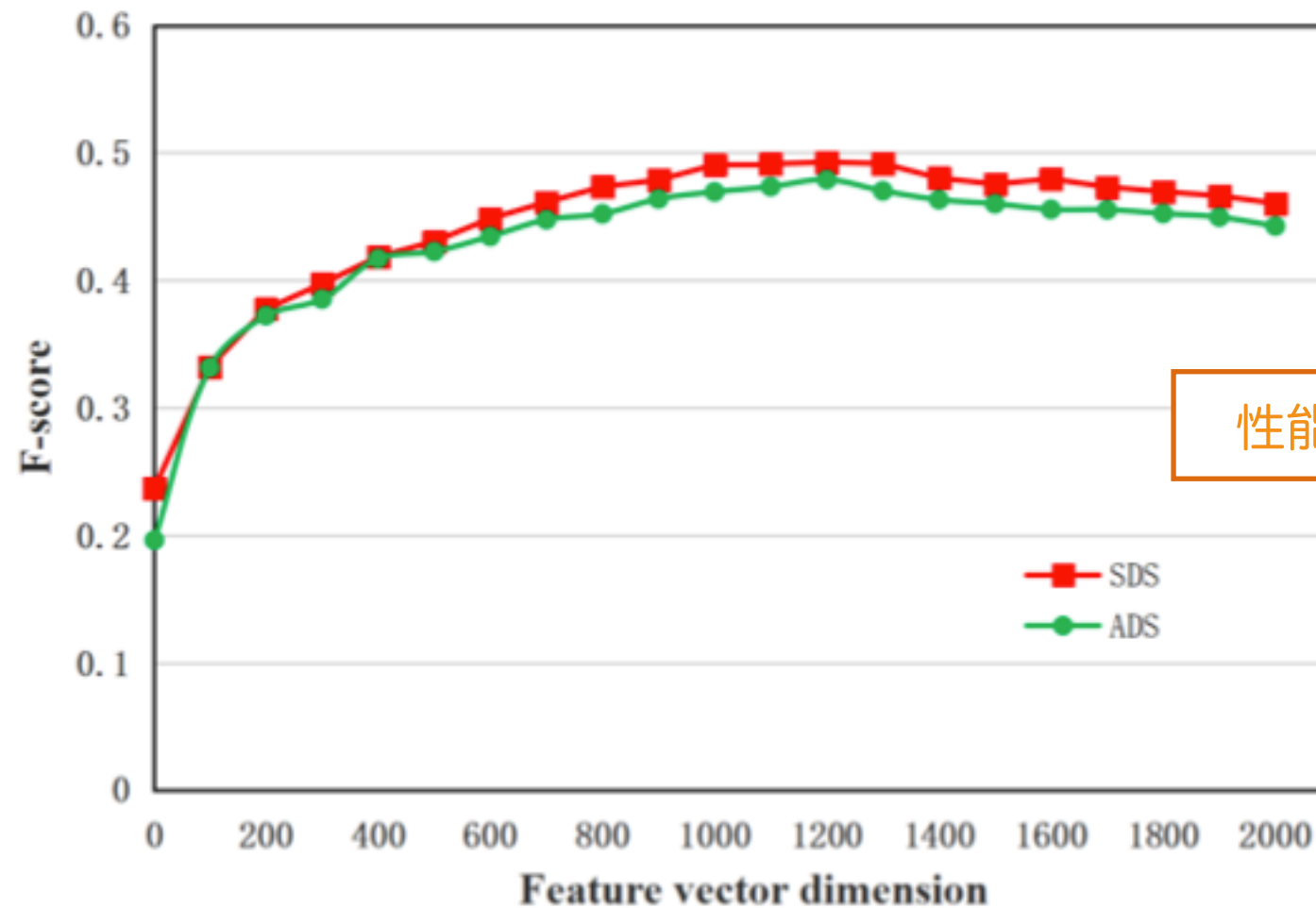


RQ3 | BugSumのパラメータが与える影響

○3つのパラメータ

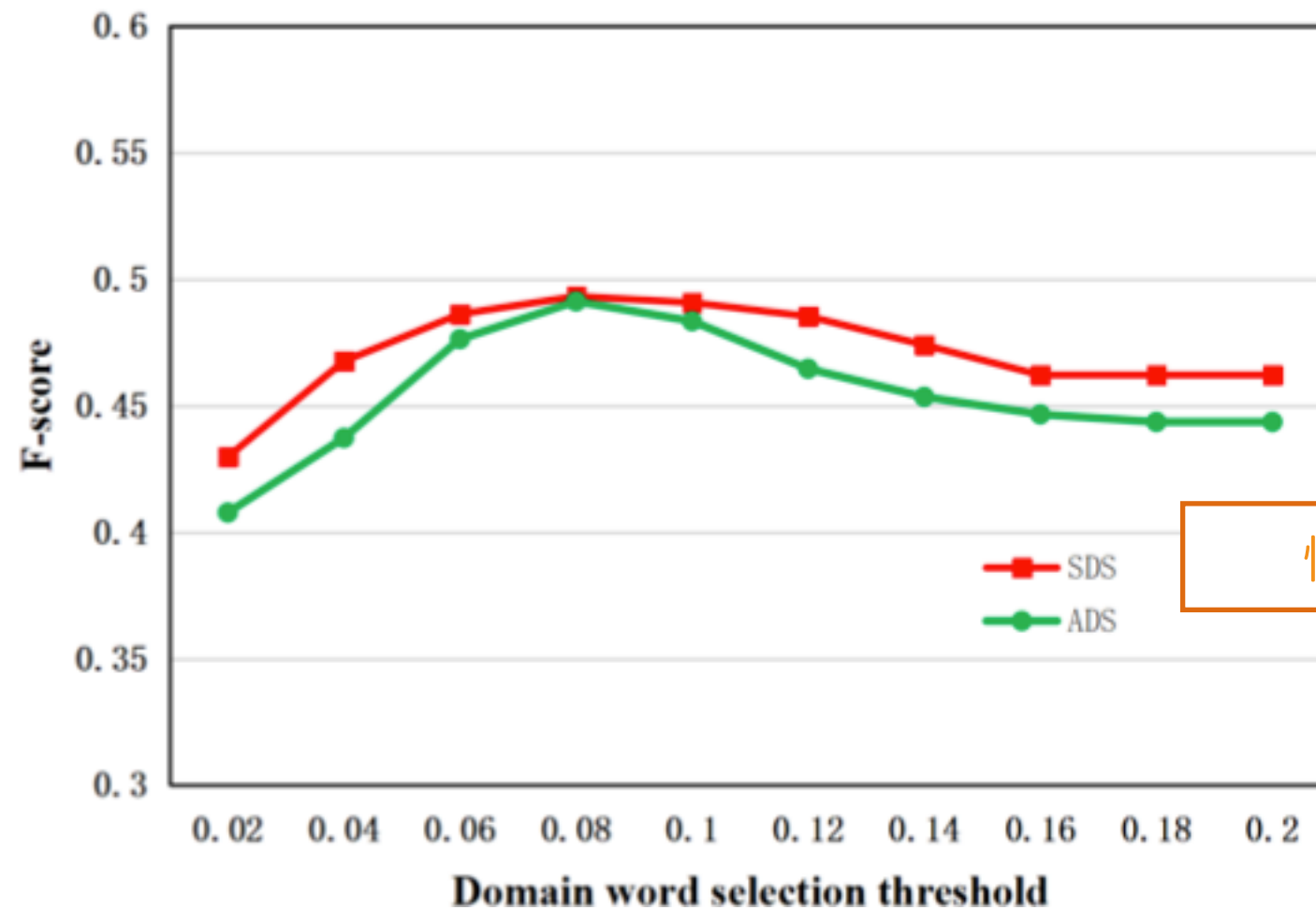
- 特徴ベクトルの次元
- 主要単語選択の閾値
- ビームサーチアルゴリズムのビームサイズ

パラメータ | 特徴ベクトル



- 200以下：性能低下
- 1000 ~ 1400：安定
- 1400以上：低下 → 情報が不十分な限られた特徴しかとれないから

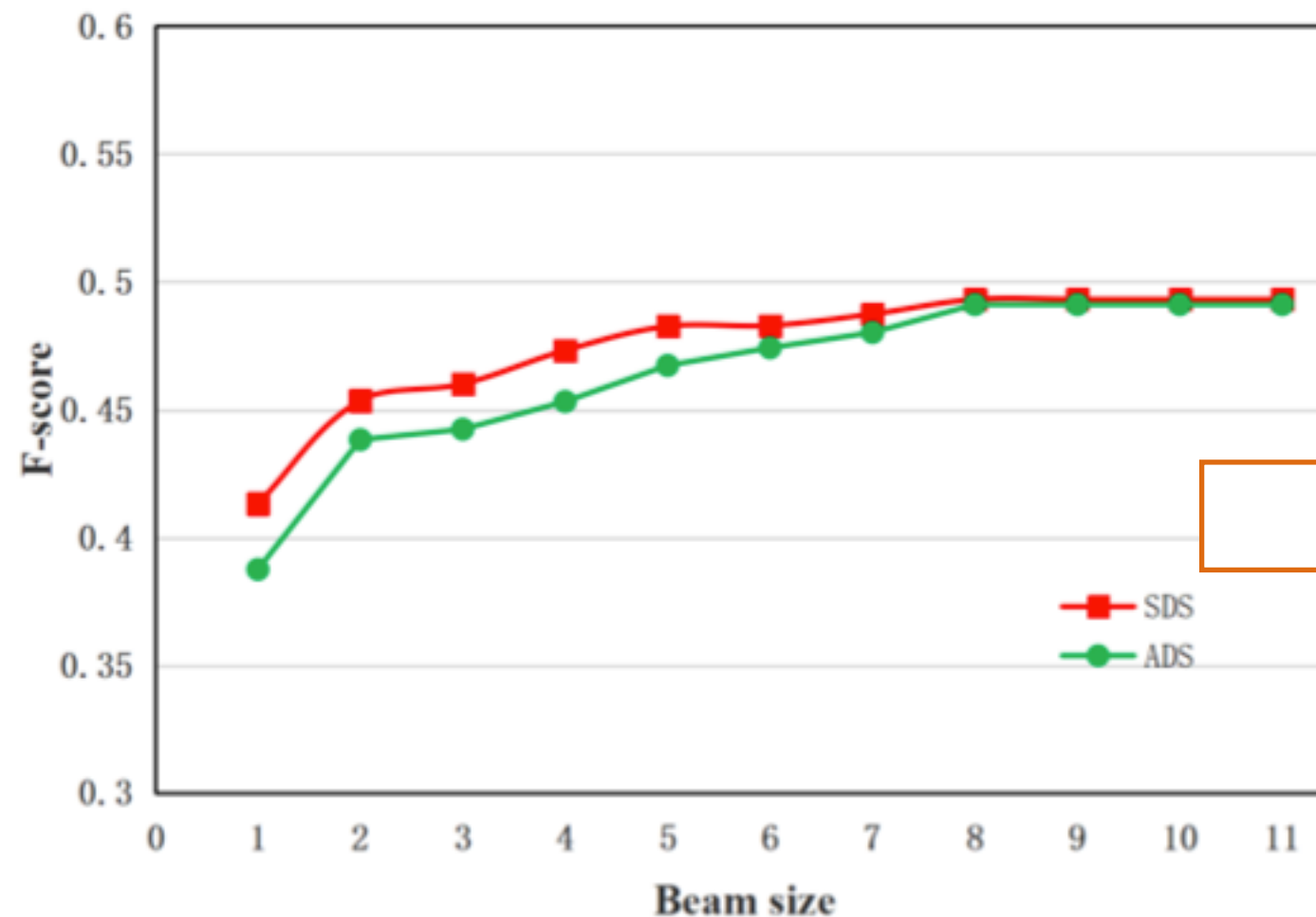
パラメータ | ドメイン単語選択の閾値 θ



性能表では0.08

- 小さすぎ → 選択されるドメイン単語が多くなる
- 大きすぎ → 選択されるドメイン単語が小さくなる

パラメータ | ビームサイズb



性能表では8

- ビーム検索アルゴリズムはb個の候補文セットを保持する
- 性能の表は8に設定 (計算時間と性能のバランスを取るため)

RQ | 文の特徴抽出と動的選択について

○ 特徴量抽出

- SFE(Sentence Feature Extraction) : 文からテキストの特徴量を抽出する
- BoW (Bag-of-Words) : 単語の頻度を保持し, 隣接する単語間の元の順序や関係性を無視

○ 選択

- SMM : 最も高いスコアを持つ文を選択する
- DS : 選択された文の包括性を向上させる (提案手法)

Dataset	Strategy	F-score	Precision	Recall	Pyramid	R-1	R-2
SDS	BoW SSM	0.297	0.411	0.241	0.307	0.436	0.092
	BoW DS	0.396	0.496	0.344	0.550	0.517	0.125
	SFE SSM	0.381	0.522	0.306	0.543	0.509	0.113
	SFE DS	0.493	0.629	0.413	0.661	0.589	0.194
ADS	BoW SSM	0.294	0.405	0.254	0.493	0.460	0.112
	BoW DS	0.386	0.451	0.353	0.562	0.512	0.207
	SFE SSM	0.377	0.467	0.322	0.528	0.487	0.180
	SFE DS	0.491	0.611	0.417	0.692	0.564	0.270

BoWは単語の頻度と文脈が取れないので精度・読みやすさが損なわれる
冗長な意味的特徴を持つ文を選択する傾向にあるためSSMはRecall低い

ベースライン

ベースライン

○ DeepSum [22]

- 事前に定義されたフィールドワードと文のタイプに当てた教師なし (報告文, コメント文, コード部分)
- 主にバグレポートの単語の頻度に基づく

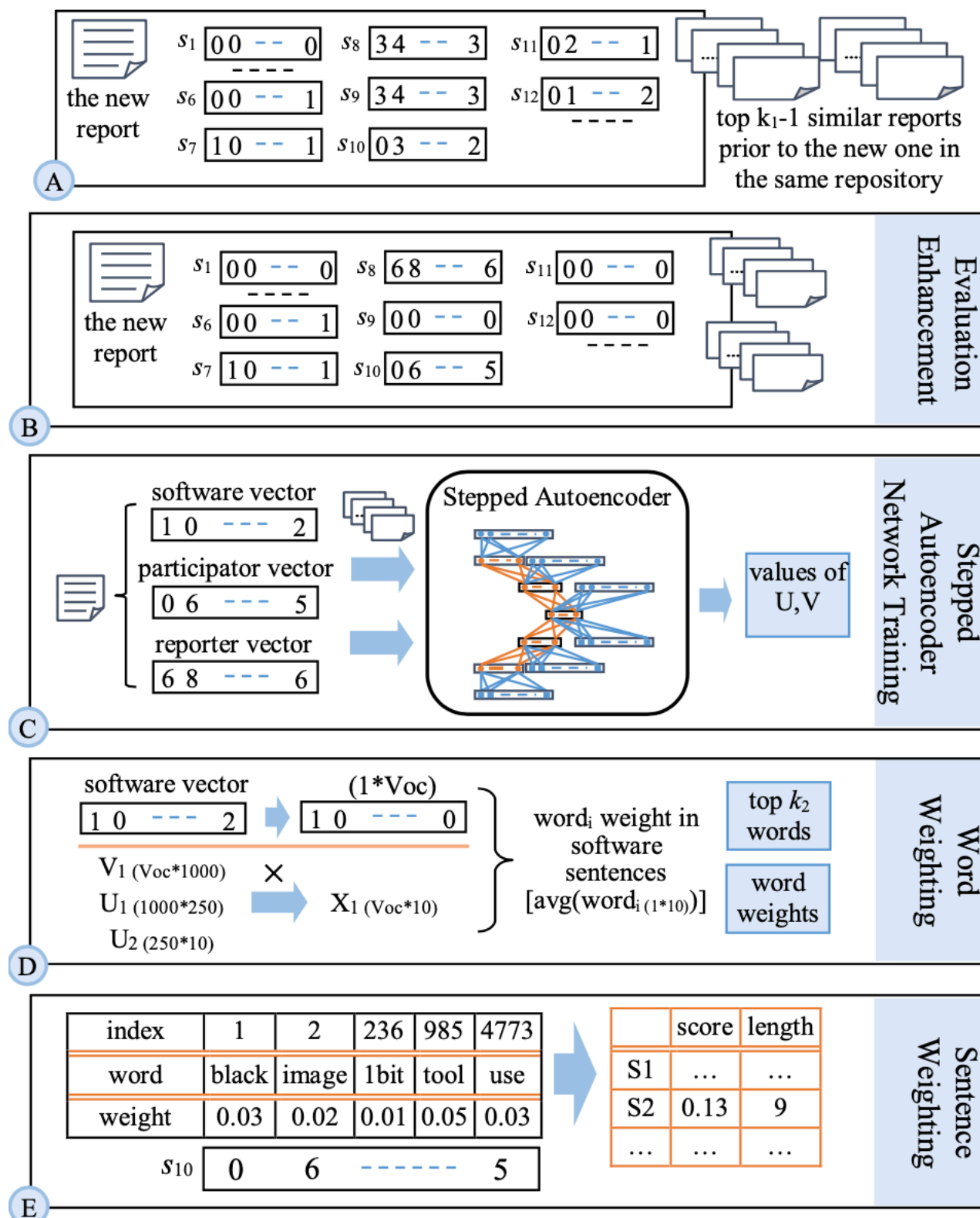
○ Grasshopper [52], DivRank [30]

- 教師なし, Noise Reducer [28]によって強化, 文脈情報に着目
- ネットワークグラフの中心性? ランク付け手法?

○ Hurried [27]

- 教師なし,
- 読者の行動をモデル化 ← マルコフ連鎖で近似
 - 他の文章に評価され, タイトルと説明に近い文章を探す
- コメント:バグに関する3種類の情報 (主張、仮説、提案) が含まれている
 - 変化していくので文脈を把握する必要がある
- ランク付け

DeepSum



類似：説明文とコメント文のテキストの単語頻度ベクトルのコサイン類似度で判定
 入力：類似した100件のバグ報告

引用部分を削除

以下の3つにベクトルをマージ
 - コード部分，参加者のテキスト，報告者のテキスト

単語 i の3つの重みを合計
 上位 k_2 個の単語を顕著な単語として選択

顕著な単語に基づいて有効な文を選択
 (長さ制限下で最大スコアのものの)

ベースライン

○ Centroid [40]

- 主にバグレポートの単語の頻度に基づく，文脈情報に着目
- 教師なし，Noise Reducer [28]によって強化

○ MMR [6]

- 教師なし，Noise Reducer [28]によって強化
- 新しさに基づいて文章を選択する

○ BRC_[43]，ACS_[17]

- 教師あり

参照

評価指標

○ 精度

- $Chosen$: 選んだ文章
- GSS : ゴールドスタンダード
- $Num_{ChosenListed}$: $Chosen$ がアノテーションによってリストアップされた回数
- $Num_{MaxListed}$: 対応する単語量制限の最大可能量
- $Recall = \frac{|Chosen \cap GSS|}{|GSS|}$
- $Precision = \frac{|Chosen \cap GSS|}{|Chosen|}$
- $Pyramid = \frac{Num_{ChosenListed}}{Num_{MaxListed}}$ [36]

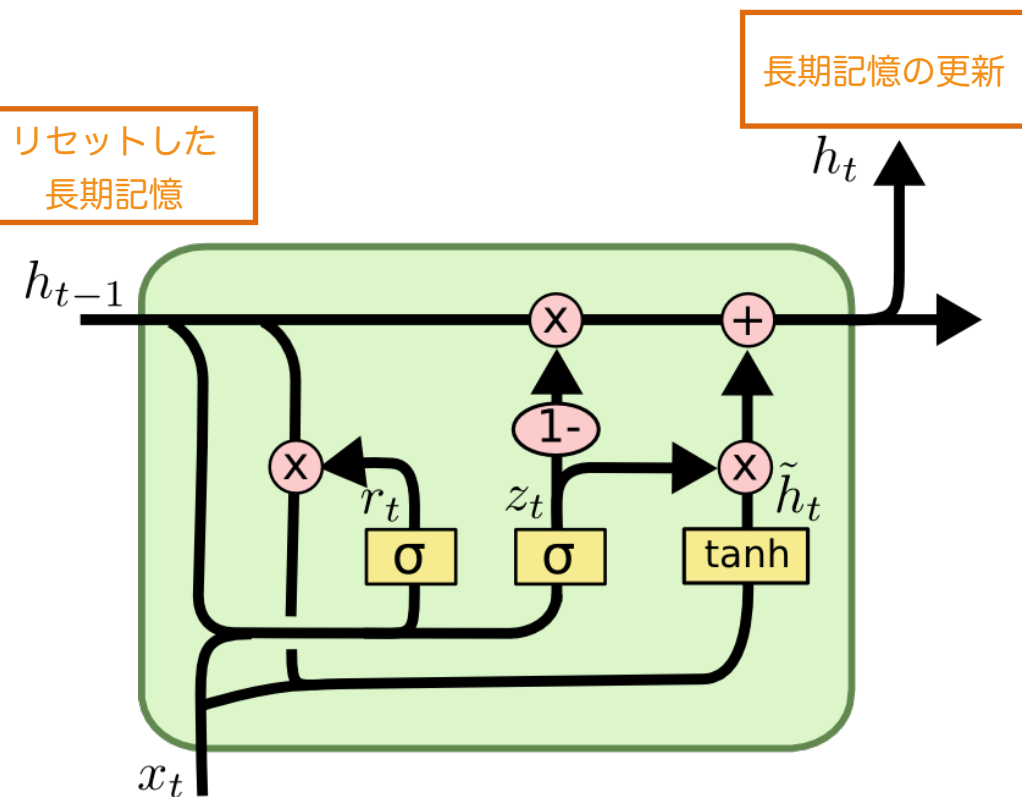
○ 読みやすさ

- Rouge-1, Rouge-2 [23]
- 要約のsumとAbsSumの間に連続的に重複するユニット数で品質評価

GRU | 手順

○ LSTMを改良

- 忘却ゲートと入力ゲートを単一の「更新ゲート」に組み合わせる
- セル状態と隠れ状態をマージし、他のいくつかの変更を加える



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

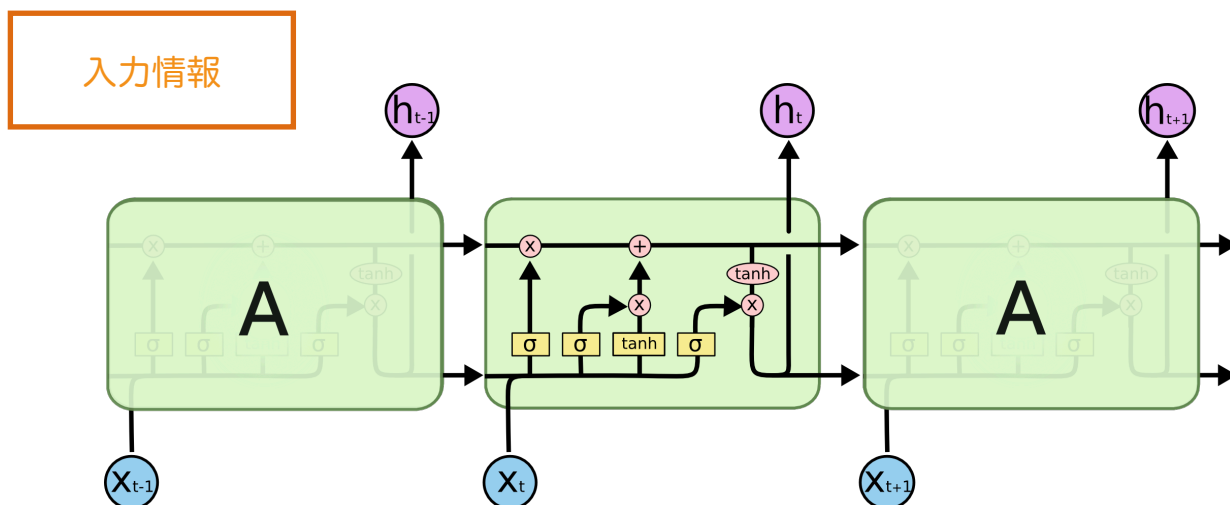
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

古い長期記憶と
現在の記憶を混合する割合

記憶率

現在の状態から
作り出された記憶

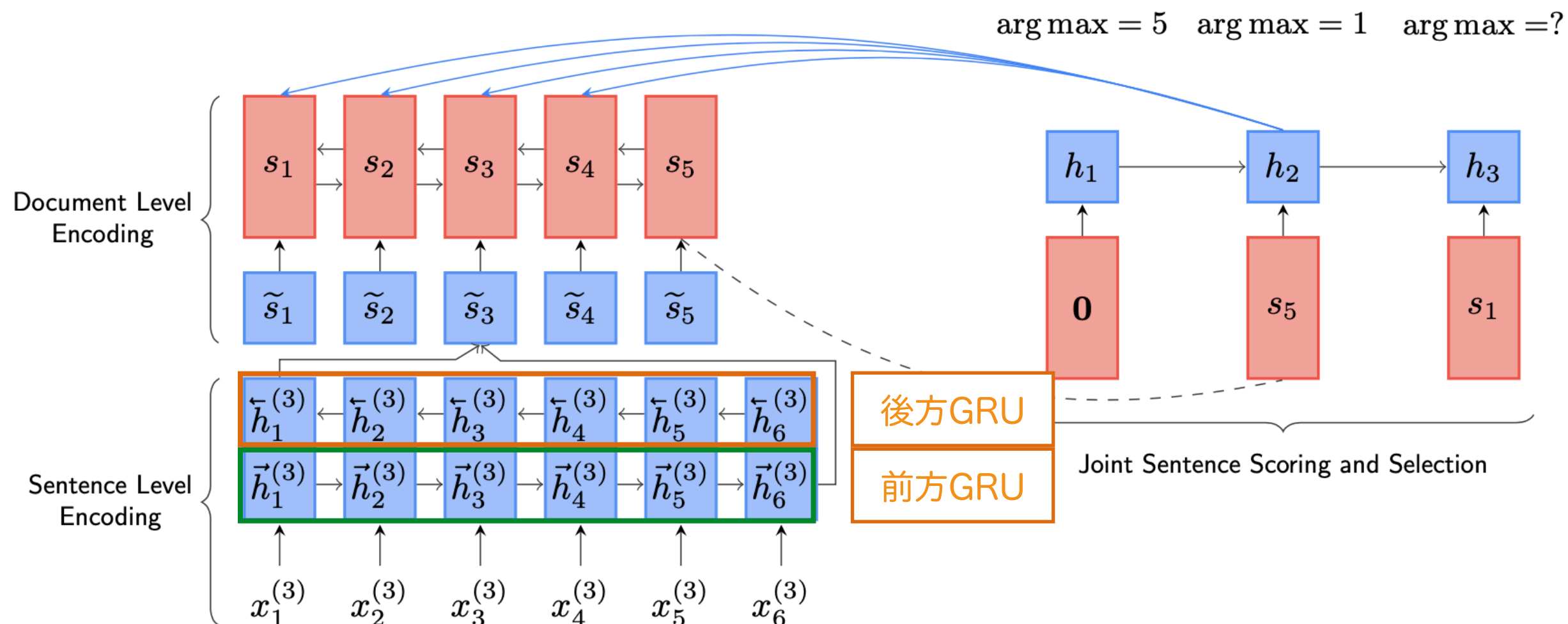
前の状態から
伝達された長期記憶



[GRU] <https://axa.biopapyrus.jp/deep-learning/rnn/gru.html>

[LSTM] <https://qiita.com/KojiOhki/items/89cd7b69a8a6239d67ca>

BI-GRU



$$z_i = \sigma(\mathbf{W}_z[x_i, h_{i-1}])$$

$$r_i = \sigma(\mathbf{W}_r[x_i, h_{i-1}])$$

$$\tilde{h}_i = \tanh(\mathbf{W}_h[x_i, r_i \odot h_{i-1}])$$

$$h_i = (1 - z_i) \odot h_{i-1} + z_i \odot \tilde{h}_i$$

古い長期記憶と
現在の記憶を混合する割合

記憶率

現在の状態から
作り出された記憶

前の状態から
伝達された長期記憶

提案手法 | 特徴量抽出

○ 双方向GRU (Bi-GRU) [51]

- 階層型オートエンコーダー→文章の特徴量を得る
- 文章を1つずつ抽出することで総和を求める
- 文章のスコアリングと選択を同時にする
- 文章の相対的な重要性を学習する

リカレントユニットには双方向GRU (Bi-GRU) [51]を採用

評価指標

○ 精度

- $Chosen$: 選んだ文章
- GSS : ゴールドスタンダード
- $Num_{ChosenListed}$: $Chosen$ がアノテータによってリストアップされた回数
- $Num_{MaxListed}$: 対応する単語量制限の最大可能量
- $Recall = \frac{|Chosen \cap GSS|}{|GSS|}$
- $Precision = \frac{|Chosen \cap GSS|}{|Chosen|}$
- $Pyramid = \frac{Num_{ChosenListed}}{Num_{MaxListed}}$ [36]
- : 複数人のアノテータがいる場合に要約の品質をよりよく測定する

○ 読みやすさ

- Rouge-1, Rouge-2 [23]
- 要約のsumとAbsSumの間に連続的に重複するユニット数で品質評価

提案手法 | 要約文の選択

- 抽出法
- バグレポートからk個の文が選択されたとする場合,
選択された文からなる集合をChosen, 特徴ベクトルの次元をdとする
- 選択された文を使ってフルテキストベクトルDFを再構成
- δ : DFとDFの間の平均二乗誤差 (MSE) を用いて再構成損失を表す
 - 小さいほど情報量が多い
- bugsum :
単語量制限 θ の下で δ を最小化する最適な文のセットを見つけることが目的