

OSSプロジェクトへのオンボーディング支援のための Good First Issue自動分類

60210105 堀口 日向

1. 背景と目的

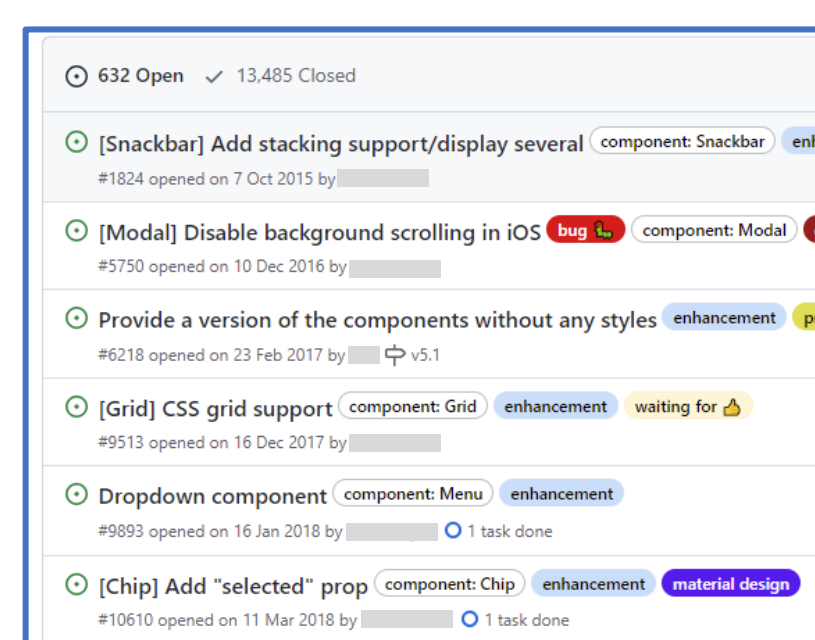
プロジェクトの持続的な開発のためには、Issueに対する開発者からの貢献が必要

不慣れなプロジェクトのIssueを理解し貢献するのに多くの時間がかかる

新規開発者



バグや機能要求などの課題 (Issue)



<https://github.com/mui-org/material-ui/issues>

一部のプロジェクトは、新規開発者向けのIssueに「Good First Issue」ラベルを付与

メンテナ

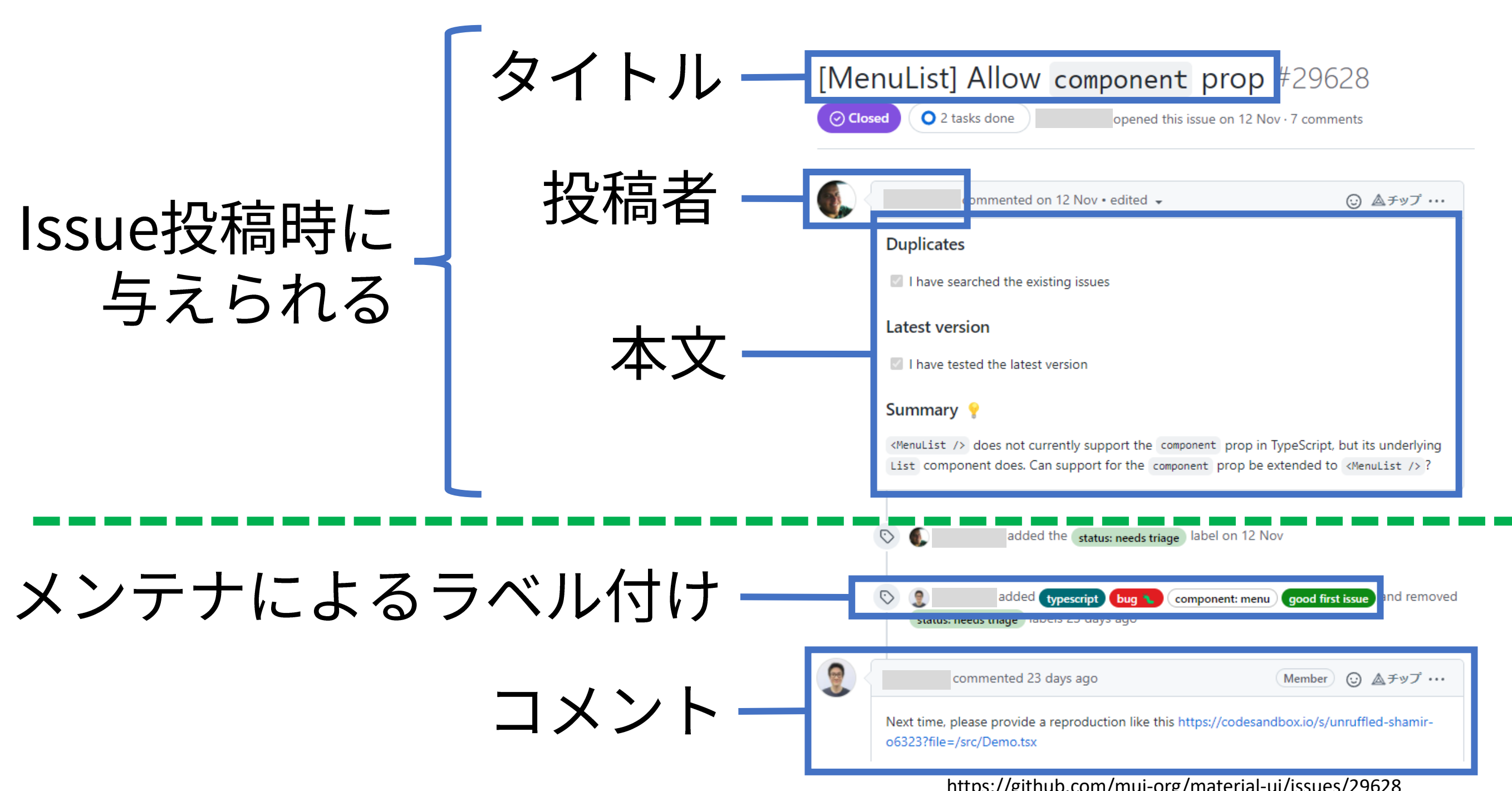


プロジェクトへの参加障壁を軽減。ただし、メンテナの負担増加

目的：GFIのラベル付けを自動化する深層学習モデルの構築

2. 方法

2.1. モデルへ入力するデータを決める



メンテナによるラベル付けの前に与えられる情報のみ利用

2.3. モデルの損失関数

- 損失関数 (Binary Cross Entropy)

$$\text{Loss}(y_i, \hat{y}_i) = -\text{weight}_i(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$$

y_i : i 番目の正解ラベル, $\hat{y}_i$: i 番目の予測値

- 損失関数の重みづけ (不均衡データ対策)

$$\text{weight}_i = \begin{cases} \frac{N}{N_{GFI}} & \text{if } i\text{th class is GFI} \\ \frac{N}{N_{Regular}} & \text{otherwise} \end{cases} \Rightarrow \text{各クラスのサンプル割合の逆数}$$

3. 評価

3.1. データセット

GitHub上のGFI数上位10プロジェクト

合計GFI数 (正例)	合計通常Issue数 (負例)
6,049 [件]	65,060 [件]

3.2. 評価方法

- 2種類の方法でモデルを学習

① プロジェクト個別のデータで学習

10分割交差検証

② クロスプロジェクトで学習

9個のプロジェクトでモデルを学習, 残り1つで評価

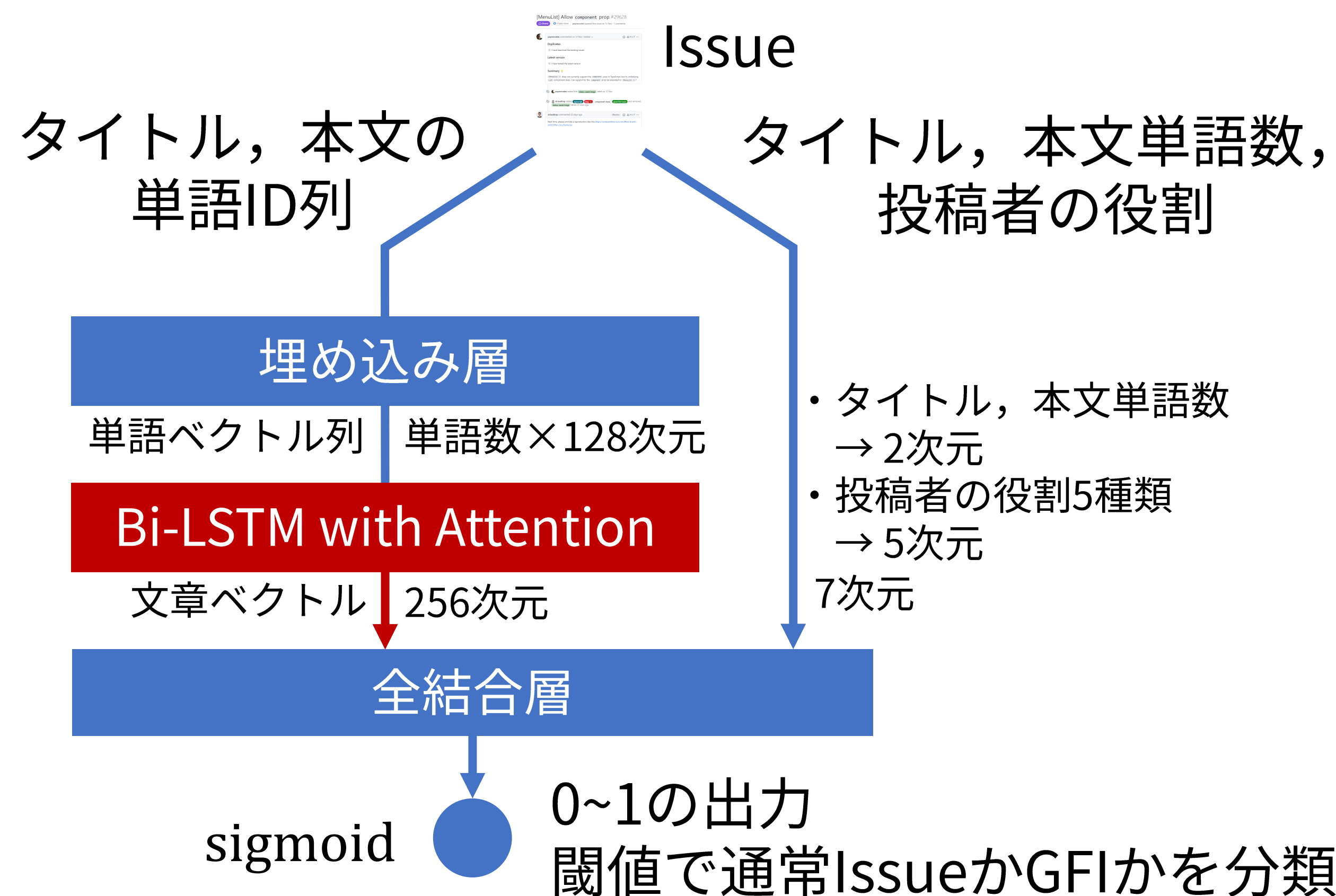
- ランダム予測モデルと比較

- 分類の閾値はF1が最大となる値を利用

4. 今後の予定

- 精度が低いプロジェクトの分析
- 誤分類したIssueの特徴を分析
 - Issueの内容
 - Attentionの重みの可視化

2.2. モデル設計



3.3. 結果

F1最大, 最小のプロジェクトと10プロジェクトの平均

① プロジェクト個別モデル

プロジェクト名	Precision	Recall	F1
scikit-learn/scikit-learn	62.50%	74.18%	67.32%
elastic/kibana	13.97%	26.04%	17.33%
平均	33.99%	50.31%	37.55%
ランダム予測の平均	13.68%	49.77%	19.55%

平均ではランダム予測より高精度。ただし、プロジェクトによって精度に大きな差

② クロスプロジェクトモデル

プロジェクト名	Precision	Recall	F1
scikit-learn/scikit-learn	41.26%	95.16%	57.56%
elastic/kibana	4.18%	22.29%	7.03%
平均	19.18%	52.43%	27.22%
ランダム予測の平均	13.68%	49.77%	19.55%

平均ではランダム予測より高精度。
個別モデルと比べてPrecisionが下がる
⇒ プロジェクトごとにGFIラベル付けの方針が異なる