

E-mail データマイニングに基づく適任 開発者の推 薦手法

福井 克法

はじめに

- 近年, ソフトウェア開発が大規模化している
- ソフトウェアを短納期で開発するには, プロジェクトに適任の開発者を割り当てることが重要

ソフトウェア開発現場での問題点

- 大規模な開発現場では、各開発者の今までに関わったプロジェクトや開発をすべて把握することは難しい
 - ▶ プロジェクトに適任の開発者を割り当てることが難しくなる
 - ▶ 企業が開発案件を提案する際に、開発者の知識や経験生かせずに案件を逃してしまう

既存研究

- ソースコードの変更履歴に着目したモジュール単位での適任開発者を推薦する手法が提案されている
 - ▶ Audris Mockusらの手法[1]
 - ▶ Romain Robbesらの手法[2]
- しかし、実際のソフトウェア開発では実装以外の開発過程も存在する

[1] Audris Mockus and James D. Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In Proceedings of the 24th International Conference on Software Engineering, ICSE '02, pp. 503–512, New York, NY, USA, 2002. ACM.

[2] Romain Robbes and David Röthlisberger. Using developer interaction data to compare expertise metrics. In Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, pp. 297–300, Piscataway, NJ, USA, 2013. IEEE Press.

本研究の目的

- 要件定義や設計などソースコードを変更しない開発者も含めたプロジェクト単位での推薦手法を提案する

既存手法

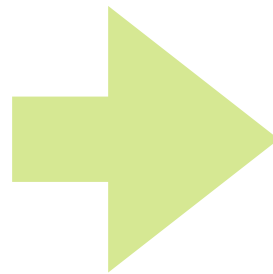
要件定義

設計

実装

テスト

運用



推薦手法

要件定義

設計

実装

テスト

運用

データソースの選択

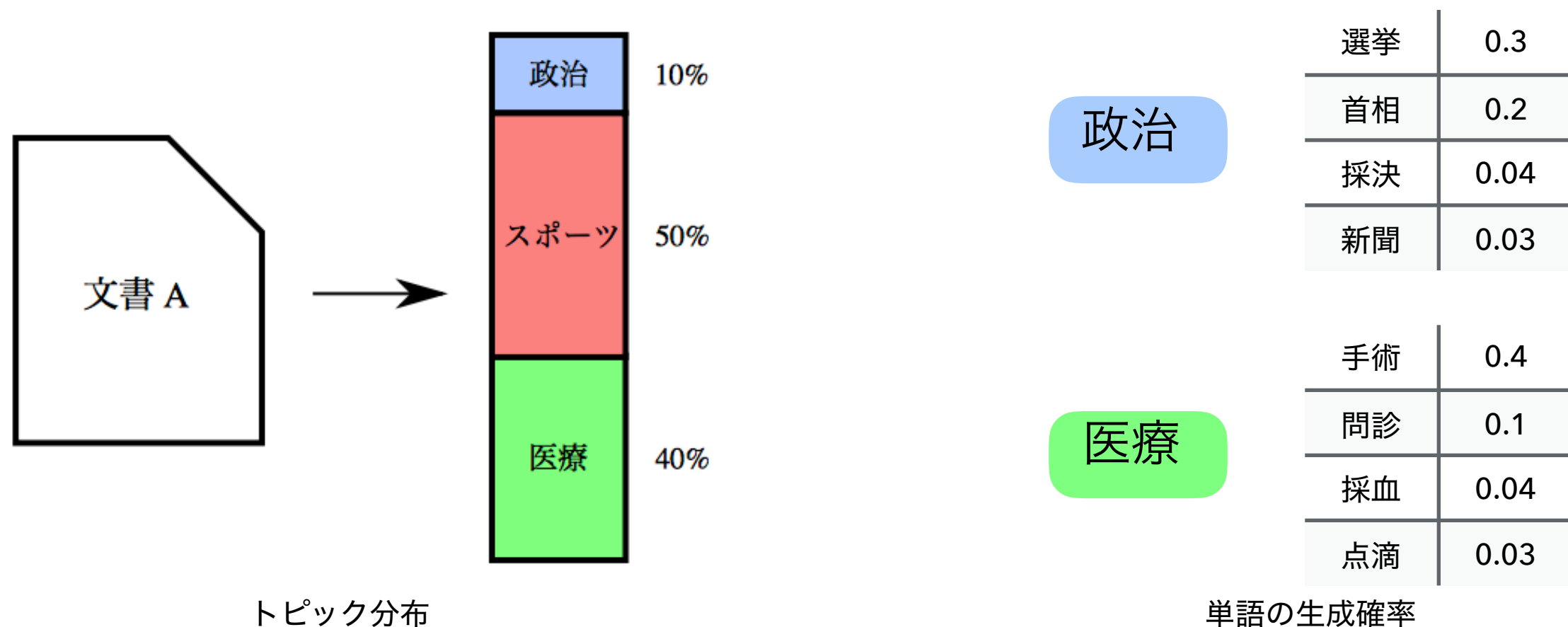
- 開発において利用されるコミュニケーションツールである**メーリングリスト**に着目する
 - ▶ メーリングリストは全工程の開発者が利用しているため、ソースコードを変更しない開発者も含めることができる
- 開発者同士のe-mailのやりとりに特に着目
 - ▶ やりとりには開発者が今まで関わった開発に関するドメイン用語が含まれていると考える
 - ドメイン用語：特定のプロジェクトで使われる特徴的な用語

メールデータのマイニング処理

- マイニング処理としてトピックモデル（LDA）を採用する
 - ▶ メールデータから、トピック（話題）を抽出することができる
 - ▶ 各メールをトピックに分類できる
 - ▶ 確率を用いて、各メールと出現する単語に重みをつけることができる

トピックモデル(LDA)

- 文章のトピックを特定することができる
 - ▶ 文章のトピック分布とトピックの単語生成確率を算出する
 - ▶ トピックごとに文章を分類できる

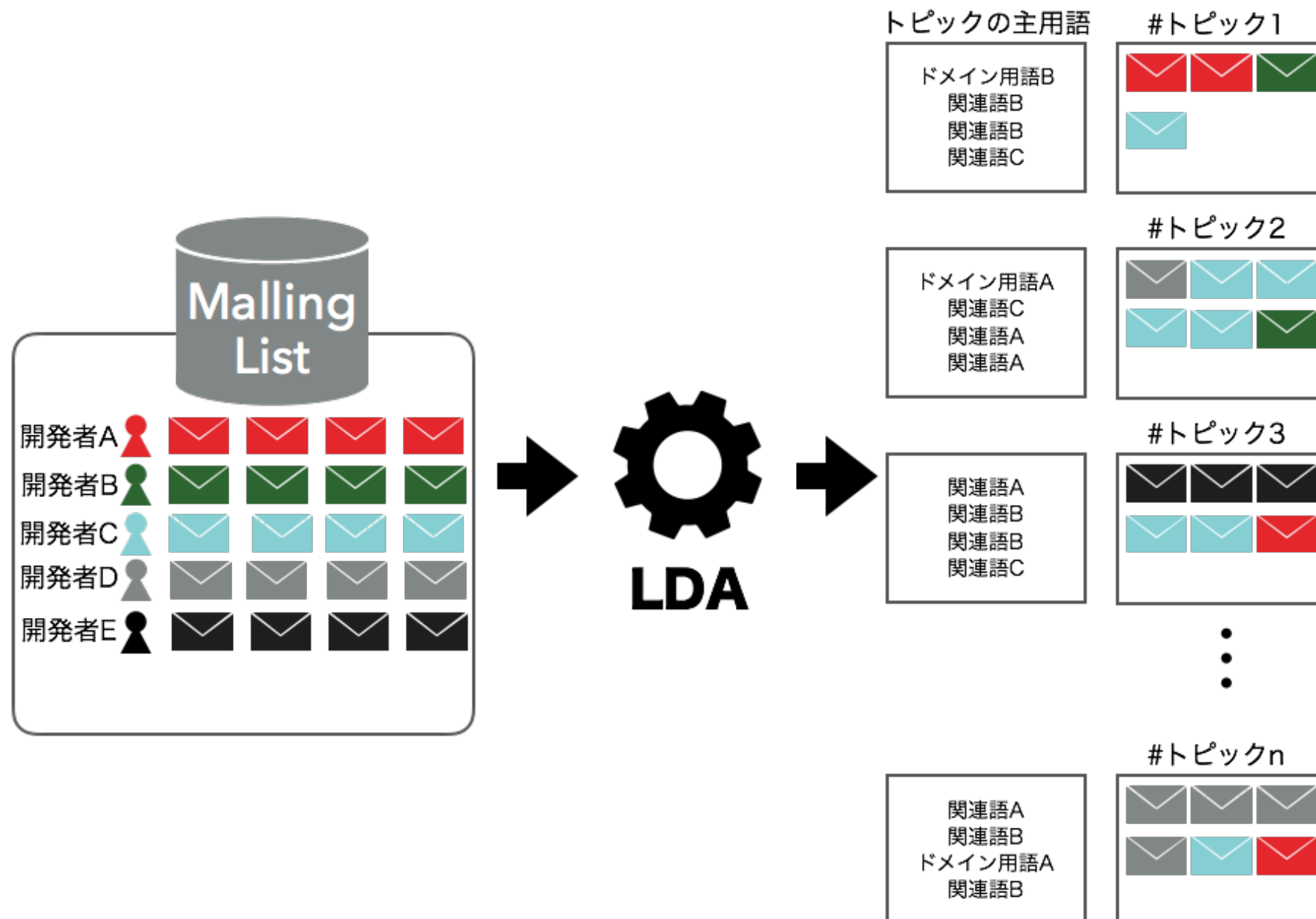


推薦手法

1. メールデータをLDAを用いてトピックに分類
2. 開発者ごとに各トピックの文章量を計算
3. 入力クエリの生成確率が高いトピックを特定
4. 開発者ごとに入力クエリと特定トピックでの文章量を用いてスコアリング
5. スコアが上位の開発者を出力

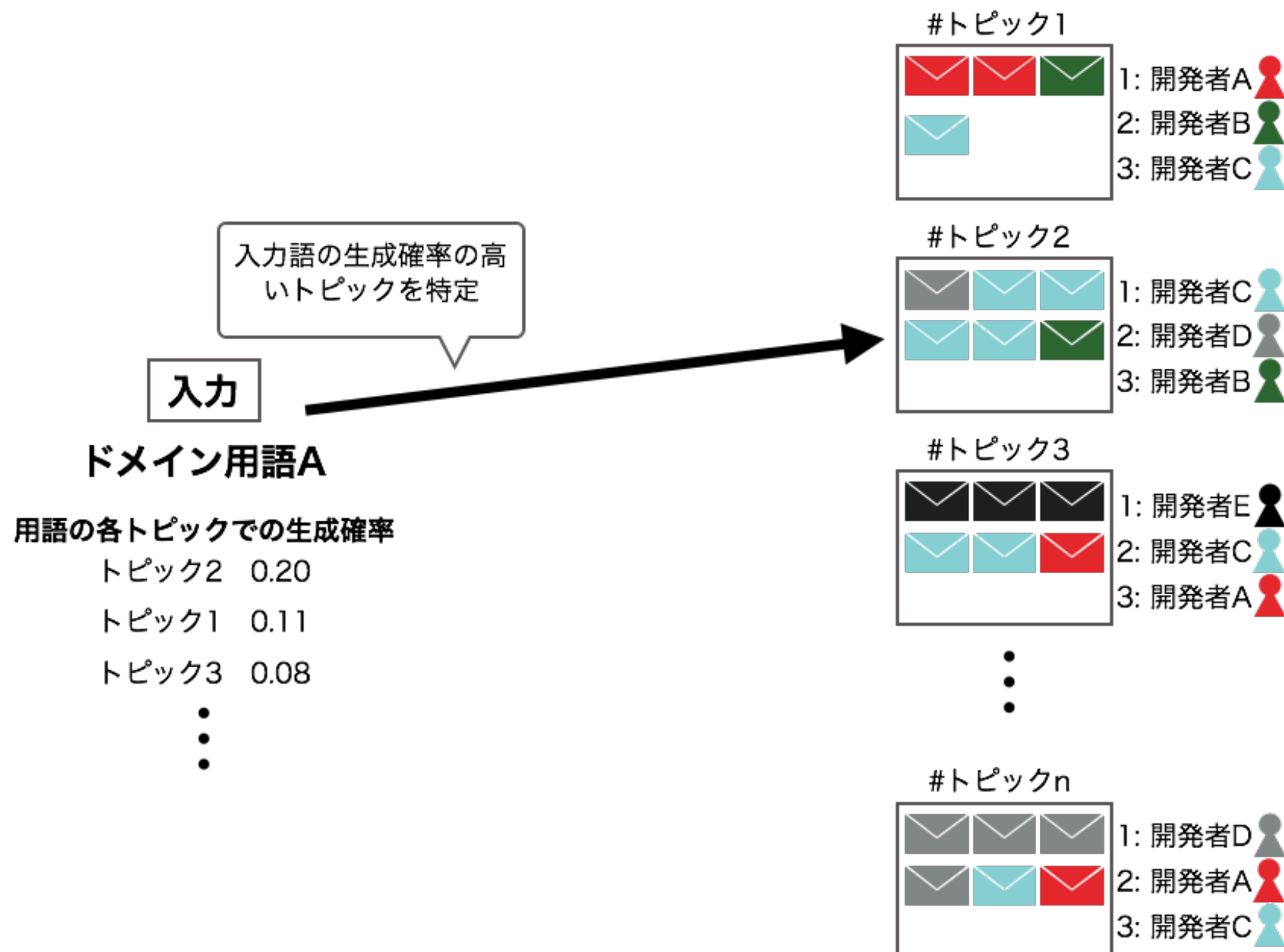
メール文章の分類

- LDAにより文章を分類しトピックごとに開発者ごとにトピックに含まれる文章量を計算



入力語からトピック特定

- 入力語の生成確率の高いトピック特定する



スコアリング方法

1. 開発者ごとに各トピックでの文章量を計算
 - ▶ **Topic_Score = 各文章のトピック分布の合計**
2. 入力クエリの確率を各トピックごとに計算
 - ▶ **Term_Score = 各トピックでの単語の生成確率**
3. 開発者ごとにスコアを計算
 - ▶ **Recommend_Score = Topic_Score * Term_Score**

手法の具体例

Proxyが生成されるトピックと生成確率
(Term_Score)

#58 0.840

#60 0.520

#2 0.020

#14 0.100

#5 0.002

開発者の文章量(Topic_Score)

	#58	#60	#2	#14	#5
devA	0.3	2.0	0.1	0.4	0.6
devB	0.1	1.0	1.0	0.2	0.3
devC	0.1	0.4	0.5	0.8	0.3
devD	0.3	0.5	1.0	2.0	1.0
devE	5.0	3.0	2.0	1.0	0.1

Proxy(サーバー用語)

各開発者でスコアを算出

dev.Aのスコア(Recommend_Score)

$$= 0.3 * 0.864 + 2.0 * 0.520 + 0.1 * 0.020 \\ + 0.4 * 0.100 + 0.6 * 0.002$$

= **1.335**

dev.Bのスコア

スコアをソートして出力

開発者	スコア
devE	5.900
devA	1.335
devD	0.734
devB	0.644
devC	0.382

評価実験

1. 複数のプロジェクトが混在しているデータに対して手法を適用
(今回は複数のOSSメーリングリストを用いていて合成データを作成)
2. 手法により推薦された開発者が実際に参加しているプロジェクトを確認し正答率を算出する

データセット

- Apacheの5プロジェクトを用いて実験を行う

プロジェクト	期間(2016年)	件数
httpd	1~3月	990件
Lucean	1~2月	943件
Maven	1~5月	902件
Openoffice	1~4月	1,120件
Spark	1~4月	1,044件
合成データ	1~5月	4,999件

前処理

- ストップワードを設定
 - ▶ 全プロジェクトで出現する733語
 - ▶ 言語処理ライブラリNLTKの標準ストップリスト
 - ▶ 頻出語から目視で追加した語

入力語の抽出

- 各プロジェクトごとに頻出語と特徴語を抽出し，入力語とする

例：httpdプロジェクト

頻出語	特徴語
mod	pid
ap	apr
apr	ap
proxy	ssl
regard	mod
pm	proxy
worker	nlm
protocol	httpd
mom	eolist

実験結果(Lucean)

- Luceanのメーリングリストでの特徴量 (tf-idf値) が高い4語

用語	上位5名の正答率	上位10名の正答率
junit	5/5	8/10
evaluate	4/5	8/10
coreloadexecutor	5/5	8/10
ementadapt	4/5	8/10

実験結果(httpd)

- httpdのメーリングリストでの特徴量 (tf-idf値) が高い
4 語

用語	上位5名の正答率	上位10名の正答率
pid (プロセス識別子)	4/5	5/10
apr(apacheportable runtime)	4/5	8/10
ssl	5/5	8/10
proxy	5/5	10/10

今後の予定

- 残りの実験
- パラメータの調整
- 入力語の見直し