リバッタルの進め方 v2

塩谷 亮太

はじめに

- この文章では，査読に対する返答の進め方を解説する
  - 「rebuttal」「author response」と言われるもの
  - 論文誌の条件付き採録でも，基本的に同様のやり方で良いと思う
- 備考：
  - この解説はコンピュータ・システム系分野の事情にかなり依存している可能性はある
- まず背景と指針について説明した後，作業の仕方を説明

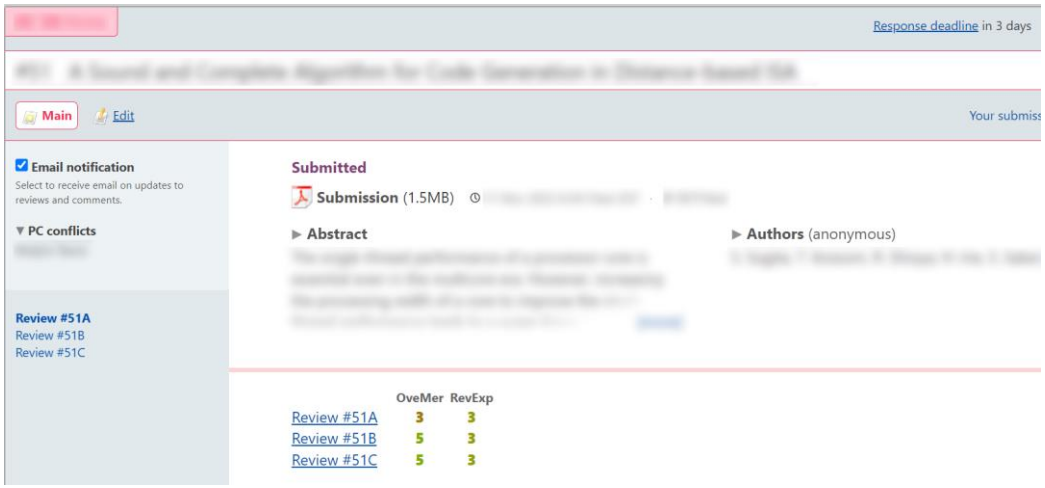
もくじ

背景と指針

前提となる査読プロセス

- コンピュータ・システム系の場合，査読者は３～６人程度
  - ◇ レベルが高い会議ほど一般には多い
- 典型的には，リバッタルがついた後に査読者同士で議論が行われる
  - ◇ リードと呼ばれる担当者が決められ，各人の意見を聞いて回る
  - ◇ ポジティブな人がリード担当になることが多いが，ランダムな場合も
  - ◇ 特に強くポジティブ or ネガティブな意見を持つ人は，詳細な意見を求められる事が多い

HotCRP 固有の事情



- HotCRP：上記画面の査読システム
  - 基本的に格が高い会議で使われることが多いように見える
  - アーキテクチャ系のトップ・カンファレンスは全部これ

- 基本的に査読コメントを書いた順にアルファベットが振られる（はず）
  - やる気がある人ほど上にくる傾向がある･･･ 気がする（偏見かも）
    - ぎりぎりに適当にやったやつは最後に入る傾向がある気がする
  - 特に正規 PC メンバーは査読数がめっちゃ多いので，上にきやすいのでは？
    - 短期間でアホみたいな数の査読をすることになる
    - 締め切り直前にまとめてやるのは無理で，かなり早めから査読がつきはじめる

基本的な指針

1. まず定性的な意見よりも，事実で反論する
   - ◇ sigcomm の指針より引用：
     「著者の回答は、意見ではなく、事実に基づいたポイントに焦点を当てることが最も効果的である」
2. 基本的には，質問で聞かれている事を答える
3. それ以外に，聞かれていない事でもこちらから説明をするのもあり
   - ◇ 共通した根底の誤解がある場合，まず最初にそれを解く文章を書くとか

取捨選択

- 大抵字数が大きく限られている事が多い
  - 全部の質問にはまともの答えられない
  - なんらかの基準でしぼらないといけない

考え方１：ポジティブな人に引っ張って貰う場合

- たとえば accept 1人が凄くポジティブ + 残り全員 weak reject の場合を想定
- ポジティブな人に引っ張ってもらう
  - ポジティブな査読者が残り査読者を説得して，引っ張り上げてくれることがある
  - この好意的な査読者をアシストしたい
    - ネガティブな人を納得させるための説得の材料を提供する

考え方２：加点にならない事は書かない？

- 明らかな弱点の場合，何を書いても無駄なことはある
  - 「～の測定がなされていない」とか言われて，それを測るのは現実的には無理な場合とか
  - なくても良い理由をくどくど書いても加点されない可能性が高い
    - 特に一般にその「～の測定」が良く行われている時とか
- 「加点にならない質問には回答しない」という指針もある
  - とある先生はこの方針だが，わりと極端な考え方であると思う
  - 不誠実な印象を与える可能性もある
- でも「なにを答えれば加点に繋がるか」という視点は重要

参考

- Sigcomm 2013 author response guidelines
  http://conferences.sigcomm.org/sigcomm/2013/misc/AuthorResponseGuidelines.pdf

  - リバッタルに対する指針や例が簡潔にまとまっていると思う
  - 引用：
    - 著者の回答は、意見ではなく、事実に基づいたポイントに焦点を当てることが最も効果的であると思われることを強調したい。
    - PCは、著者が主張する新しい結果や改善点を無視することができます。
    - また、著者の回答が投稿時の誤りを修正するものであったとしても、いくつかの誤りを含む論文を受理するかどうかの最終判断はPCが行います。

主張の整理

おおまかな流れ

1. 査読コメントの振り分け
    1. 査読コメント内の主張ごとに ID をふる
    2. 同様の主張のグルーピング
2. 主張の整理
    1. 言われている主張の要点を自分たちでまとめなおす

作業の進め方

- 共同編集可能な markdown 上で進めるのがおすすめ
  - ここでは overleaf の上に .md ファイルを置くことを想定
    - overleaf は .md を認識してシンタックス・ハイライトできる
  - 他のサービスでも別によい
    - Google document とか
    - 1 人で進めるならローカルの .md ファイルでも
- この作業に git やチケット管理システムはあまり向かない（と思う）
  - 後のタスク・スケジューリングの項で理由を説明

査読コメントの振り分け

- まず，コメントで言われている内容ごとに ID を振る

- ID のフォーマット： $<査読者><シリアル>-<タイプ>
  - 先頭に $ をつける
  - <査読者> は査読者 A や B などのアルファベット
  - <シリアル> は一意な識別番号
    - 査読者ごとには番号は被っても良い
      - A1 と B1 があってもよい
    - 査読者が同じでもタイプが異なる場合は被ってはいけない
      - A1-w と A1-c が同時にあってはいけない
  - <タイプ> は week point なら w，質問なら q，コメントなら c

- 例：
  - $A1-c, $B3-w, $E7-q など

ID フォーマットがこうなってる理由

- 主に識別を容易にするため：
  - 議論するとき「査読者 A が言ってた～に関する説明がないというやつなんだけど」よりも 「コメント A4 なんだけど」の方が早い
- タイプ部分（$A4-w の –w の部分）は無視しても識別可能な方がよい
  - 査読者ごとに一意な番号をふる
  - A1-w と A1-q が並列に存在すると，どっちかわからなくなる
- タイプ部分は対応を考える際の重要度を簡易に表す
  - ただのコメント（c）と，弱点としてあげられている点（w）では異なる
- $ はファイル内の検索性を上げるためにある
  - A1 等だと，検索した際に査読コメント内の地の文でもヒットする可能性が結構高い

ID を振った主張にタイトルをつける

- 主張ごとにタイトルをつけ，先頭に括弧とともに ID をつける
- 例：
  - ($A1-q) 関数がループで繰り返される場合はうまく学習できるのか？
  - ($A2-q) Figure 17 の Distance はどのように定義したのか？
  - ($B11-c) Distance の動的な変更をしたほうが良いか

マークダウンのセクションにまとめる

- セクションごとに主張をまとめる
  - ID+タイトル を見出しに
  - ここではレベル２見出し（##）を使用
- このとき，同様のことを言っているものをグルーピングする
  - 以下の例の場合は $A1-q, $A7-w, $F5-w がまとめられている

($A1-q,$A7-w,$F5-w) 複雑な構造をうまく学習できるのか？

* ($A1-q) How do you handle loops? The FIFO would just contain the BBL of foo over and over again. This is the main reason why ...
* ($A7-w) Unclear how the approach handles looped function calls
* ($F5-w) Unclear how recursive function calls will be handled.

主張の整理

- セクション内に以下などの情報をさらにまとめていく
  - 対応する査読コメント全文
    - 日本語にする時は必ず原文も載せる
  - 査読者の主張の要点
  - 想定される疑問や，査読者の誤解
  - 対策や回答の案など

($A1-q,$A7-w,$F5-w) 複雑な構造をうまく学習できるのか？

* ($A1-q) How do you handle loops? The FIFO would just contain the BBL of foo over and over again. This is the main reason why ...
* ($A7-w) Unclear how the approach handles looped function calls
* ($F5-w) Unclear how recursive function calls will be handled.

* A の疑問：
  * ...
* F の誤解：
  * ...

* 回答の原稿：
  * ...

タスク管理

タスク管理

- なんらかの方法でタスク管理をした方がよい
  - ◇ やるべき事が多岐にわたる
  - ◇ なにが終わってなにが終わっていないかわからなくなる

テキストによる軽量なタスク管理

- チケット管理システムや github の issue などはあまり向かない
  - 「チケット」にあたるもの自体が頻繁に変わる
    - グルーピングやまとめの際の解釈でコロコロ項目が変化する
  - そのたびにチケットを切り直したり issue を立てるのは手間が大きすぎる

- テキスト上で同様のことを軽量に行う
  - この方法はスケールしないが，査読対応ぐらいの規模なら十分
    - 論理的にはチケット管理と全く同じ事をしている
  - 柔軟性は最高にある
    - 査読コメントへの対応に特化させることもできる
  - 項目間の移動はファイル内の検索で行う

Markdown 上でのタスク管理のフォーマット

- 「TODO」の下に「おわった」「着手中」「まだ」「保留」のセクションを作る
  - おわった：対応の検討や回答原稿がおわったもの
  - 着手中：　着手が開始されたもの
  - まだ：　　まだ着手されていないもの
  - 保留：　　検討の結果，保留にすることにしたもの

```
# TODO

## おわった
* (小泉) ($A6-w,$A16-c) 既存手法に対するcontributionが不明瞭
* ...

## 着手中
* (中村) ($D8-c) late の遅れぐあいの内訳を明らかにする
* ...

## まだ
* (出川) ($A1-q,$A7-w,$F5-w) 複雑な構造をうまく学習できるのか？
* ...

## 保留
* ($B3-q) Figure 4 の曲線はなにか？ (連続するデータをとっているのか？補完した点なのか？)
    * (中村) あまりにもクソリプだと思う
```

箇条書きの項目

- 各セクションに以下を箇条書きで記述
  - 担当者と ID，タイトル
    - 担当者は決まるまでは空欄でもよい
  - これら以外の付加的な情報はここには極力書かない
    - ここに色々メモを書きたくなるが，膨らんで破綻しがち
    - 後述の詳細セクションにそう言うのは書く
- 状況ごとに箇条書きのアイテムを移動させる

```
# TODO

## おわった
* (小泉) ($A6-w,$A16-c) 既存手法に対するcontributionが不明瞭
* ...

## 着手中
* (中村) ($D8-c) late の遅れぐあいの内訳を明らかにする
* ...

## まだ
* (出川) ($A1-q,$A7-w,$F5-w) 複雑な構造をうまく学習できるのか？
* ...

## 保留
* ($B3-q) Figure 4 の曲線はなにか？ (連続するデータをとっているのか？補完した点なのか？)
    * (中村) あまりにもクソリプだと思う
```

詳細のセクションをさらに後ろにつける

- 「主張の整理」でまとめた詳細を TODO の後ろに置く
  - 細かい詳細はぜんぶそっちに書く
  - 移動は ID をファイル内検索して行うと良い

```
# TODO

## おわった
* (小泉) ($A6-w,$A16-c) 既存手法に対するcontributionが不明瞭
* ...

## 着手中
* (中村) ($D8-c) late の遅れぐあいの内訳を明らかにする
* ...

## まだ
* (出川) ($A1-q,$A7-w,$F5-w) 複雑な構造をうまく学習できるのか？
* ...

## 保留
* ($B3-q) Figure 4 の曲線はなにか？ (連続するデータをとっているのか？補完した点なのか？)
    * (中村) あまりにもクソリプだと思う


# 詳細

## (出川) ($A1-q,$A7-w,$F5-w) 複雑な構造をうまく学習できるのか？
* ($A1-q) How do you handle loops? The FIFO would just contain the BBL of foo over and over again. This is the main reason why ...
* ($A7-w) Unclear how the approach handles looped function calls
* ($F5-w) Unclear how recursive function calls will be handled.
...

## (小泉) ($A6-w,$A16-c) 既存手法に対するcontributionが不明瞭
...
```