A Crowdsourceable Protocol for Collecting User-Generated Counter-Arguments

Paul Reisert[†] Naoya Inoue[†,‡] Kentaro Inui[†,‡]

[†]RIKEN Center for Advanced Intelligence Project (AIP) [‡]Tohoku University
paul.reisert@riken.jp
{naoya-i, inui}@ecei.tohoku.ac.jp

1 Introduction

Automatic essay scoring (AES) is the task of automatically evaluating a wide-range of criteria of an essay in a pedagogical context. Such criteria includes the organization [10], thesis clarity [11] and author stance [12], to name a few. Several works have also integrated argumentative features [13, 3, 8] as a means of evaluating the overall quality of an essay. Applications such as Grammarly[1] and eRater[2] have received wide attention for their ability to automatically assess the contents of an essay.

While several studies exists for evaluating the criteria of an essay, they do not provide useful feedback to the writer which in turn could improve their critical thinking skills. An example of the usefulness of constructive feedback is shown in Figure 1. In response to the *prompt* P_1 (i.e., *Are police too willing to use force?*), consider the following argument A_1 extracted from a student's essay: *Police are too willing to use force. Police are using excessive force all over the US and its not recorded.* In response to this argument, a teacher would provide constructive feedback to the student for improving their argument (e.g., CF_3:"Not all actions of the police are violent"). Afterwards, a student could revise their argument to produce a stronger argument (i.e., R_1) and in turn learn about how to produce stronger arguments.

Toward improving critical thinking skills for writers, recent works have emphasized the importance of both fallacies and counter-arguments. [4] created a mobile game which allowed users to identify fallacies in arguments. In the pedagogical context, several studies have worked towards identifying common fallacies in student essays [9, 7, 2]. For counter-arguments, [14] created a task for retrieving the best counter-argument for a given argument. [6] generated counter-arguments by extracting evidence to an argument via Wikipedia. However, with a lack of corpora for modeling constructive feedback, it remains an open issue

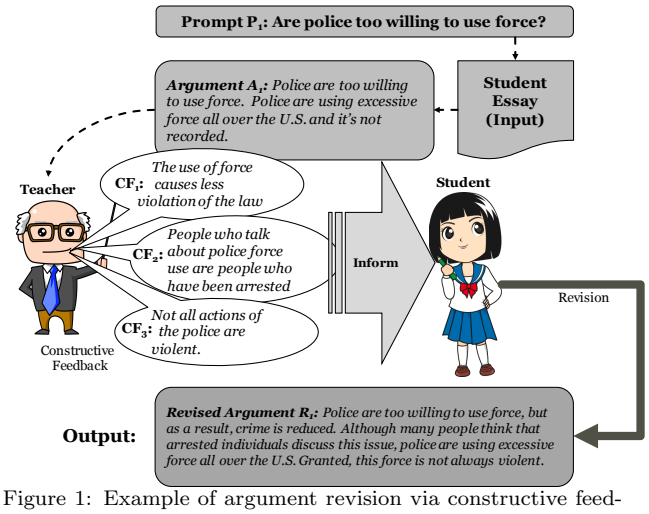

Figure 1: Example of argument revision via constructive feedback.

as to how to automatically generate constructive feedback useful for improving one's critical thinking skills.

There are many benefits for automatically generating constructive feedback. First, as mentioned *a priori*, generating counter-arguments will be useful for automatically assessing the content of an essay and providing instant feedback to students. This will allow students to produce high-quality arguments while simultaneously improving their critical thinking skills. Students will also be able to learn about what types of arguments they struggle with most. Additionally, the time spent producing critical comments will be reduced for teachers, enabling them to select the best critical comments for a students argument. Second, the generation of counter-arguments will be useful in a debate setting, where participants are engaged in a debate with machines (e.g., IBM Project Debater[3]).

There are several challenges for creating a corpus useful for modeling constructive feedback. First, there exists several hundred fallacy types in the wild [1], such as *ad hominem*, *begging the question*, and *appeal to authority*. Second, without knowing the fallacy type in advance, it can be difficult to generate mean-

[1] https://www.grammarly.com/
[2] https://www.ets.org/erater
[3] https://www.research.ibm.com/artificial-intelligence/project-debater/

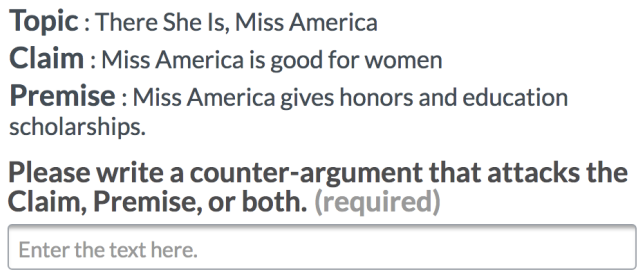

Figure 2: Interface for the Counter-Argument Generation Stage

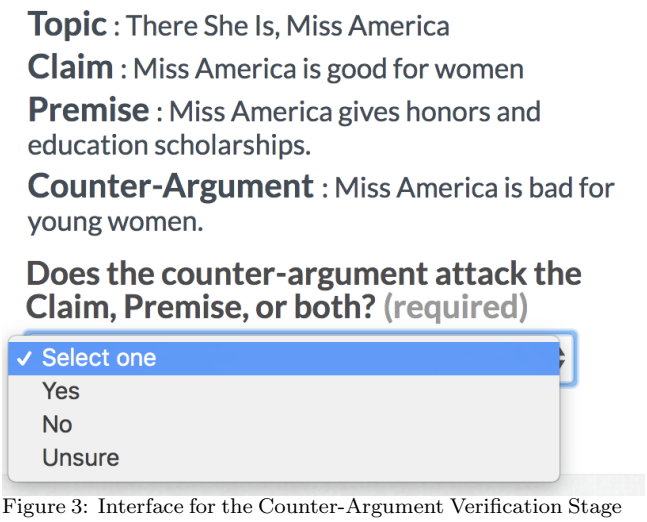

Figure 3: Interface for the Counter-Argument Verification Stage

ingful feedback. Third, the annotation of fallacies is costly, time-consuming, and not guaranteed to be high quality. Therefore, it is necessary to compose an efficient method for constructing a high-quality, large-scale corpus for constructive feedback generation.

In this work, we conduct two parallel crowdsourcing tasks in order to determine if a large-scale, high-quality corpus of user-generated counter-arguments needed for modeling constructive feedback can be created. We first instruct non-expert workers to produce counter-arguments simply given an argument. Simultaneously, we conduct another crowdsourcing experiment which instructs workers to produce a counter-argument after identifying a specified fallacy type. Our results suggests that non-expert annotators can produce useful counter-arguments, especially when first instructed to identify a fallacy type.

2 Corpus of counter-arguments

In this section, we describe our proposed method for collecting user-generated counter-arguments.

2.1 Data

We experiment on top of the Argument Reasoning Comprehension (ARC) corpus [5]. The ARC corpus contains 2,477 context-independent arguments consisting of a topic, claim, premise, warrant, and anti-warrant pairs created by crowdsourcing, where the warrant links the premise and claim and the anti-warrant does not. In total, there are roughly 50 diverse topics.

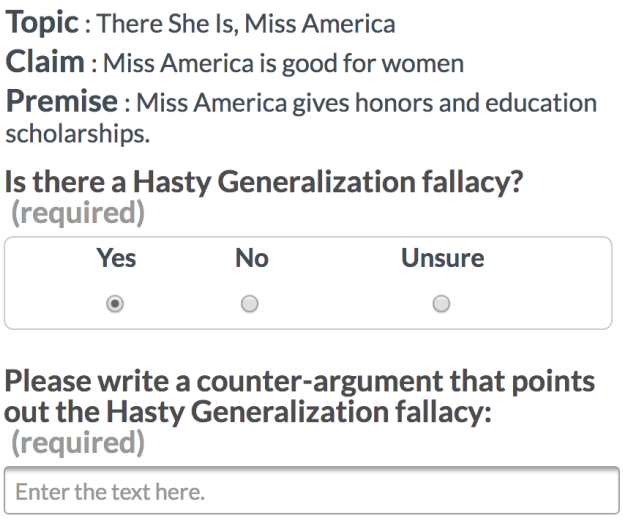

Figure 4: Interface for identifying a fallacy and producing a counter-argument

2.2 Crowdsourcing

We use the crowdsourcing platform Figure Eight.[4] Our assumption is that a large-scale corpus of counter-arguments can be constructed by non-expert crowdsourcing annotators with appropriate guidelines and examples.

2.2.1 Counter-argument generation without fallacy identification (CAG)

Towards creating a large-scale corpus of user-generated counter-arguments, we first conduct several trial experiments on Figure Eight for collecting counter-arguments. We immediately discovered that the most important setting is the minimum time per instance, which prevents workers from tainting the input (e.g., quickly copying and pasting parts of the argument). Other problems included counter-arguments of the same stance as the claim/premise and candidates in a non-English language. Thus, we update our guidelines accordingly. Our final settings consists of the following: minimum time of 10 seconds per instance, level 3 annotators (i.e., highest quality annotators), and $0.10 per candidate answer.

The final interface at this stage is shown in Figure 2. Per given *topic*, the worker is shown the *claim* and *premise* and instructed to produce a counter-argument that attacks one or both of them.[5] Each

[4] http://www.figure-eight.com

[5] We adopt this approach from [14], which states that a counter-argument can attack one or both components.

worker was instructed to produce a counter-argument that is one sentence and in English.

To verify whether the produced counter-argument candidates function as a counter-argument, we conduct an additional crowdsourcing job. For each candidate, we ask workers to verify whether it functions as a counter-argument. For controlling the reliability of each worker, we created 8 test questions using a development set. The test questions were randomly inserted into the task. Workers had to maintain an accuracy of over 70% on the test questions; otherwise, they were immediately ejected from the task. For this task, each worker was awarded with $0.05 per answer. In total, each candidate was judged by 5 workers.

For the verification stage, workers were given the *topic*, *claim*, *premise*, and additionally the candidate (see Figure 3). The workers were instructed to select *yes* if the candidate functioned as a counter-argument; otherwise, the worker selected *no*. If the workers were unsure about whether the candidate functions as a counter-argument, the workers selected *unsure*.

2.2.2 Counter-argument generation with fallacy identification (CAG-F)

In addition to our crowdsourcing experiment in Section 2.2.1, we conduct an additional study in which crowdworkers were asked to identify a pre-specified fallacy type. In total, we randomly select the following 5 fallacy types and their examples from SoftSchools[6]:

- **hasty generalization** A fallacy in which a person hastily assumes that something is generally always the case based on a few instances.
- **red herring** A fallacy in which a person changes the subject to take attention away from the original argument.
- **questionable cause** A fallacy in which a person incorrectly believes that something is the cause for something else.
- **begging the question** A fallacy in which the premise repeats the same information as the claim.
- **appeal to common practice** A fallacy in which someone believes something is acceptable only because most people do it.

To simplify the task for the crowdworkers, we create 5 separate jobs where each worker is asked if an argument contains a fallacy of the specified type. If so, each worker was instructed to produce a counter-argument. An example of the interface for the *Hasty Generalization* fallacy identification is shown in Figure 4.

Because it is time-consuming to manually create gold data for positive instances of one fallacy type, and we cannot guarantee that every fallacy type will be in the dataset, we do not create test questions for our 5 crowdsourcing jobs. Instead, at this stage, we rely on the majority vote of workers for fallacy identification. At a later stage, we will use the results of our experiment for creating gold data which can help control the reliability of future annotations. Simultaneously, due to the limited amount of produced counter-arguments, we avoid integrating a stage for verifying the counter-argument candidates and manually analyze them.

[6] http://www.softschools.com/examples/fallacies/

Fallacy Type	1/5	2/5	3/5	4/5	5/5
Appeal to Common Practice	0	4	3	0	0
Begging the Question	0	7	1	0	0
Hasty Generalization	0	8	0	1	0
Questionable Cause	0	6	3	1	0
Red Herring	0	4	10	1	0

Table 1: Distribution of votes for positively-identified fallacies.

3 Analysis and Discussion

For CAG, we generate 2,625 counter-arguments for 525 arguments and verify each using 5 annotators. After majority voting, we find that roughly 68.3% of the counter-arguments were categorized as *yes*, 31.5% *no*, and the remaining were *unsure*.

For each of the 5 jobs in CAG-F, we had 5 annotators judge 100 instances. We first collect all positive instances of fallacies identified in CAG-F with an agreement of 3/5 annotators or more. In total, we collect 20 instances. The distribution of votes for positively-identified fallacies (i.e., annotators said *yes* for a specific fallacy type) is identified in Table 1. We note that no fallacy types had perfect agreement. We believe this will improve with the integration of test questions.

Stage	Useful (%)	Useless	Undecided	Total
CAG	14 (0.19)	40 (0.54)	20 (.27)	74
CAG-F	16 (0.38)	23 (0.55)	3 (0.07)	42

Table 2: Usefulness of Counter-Arguments for CAG and CAG-F.

For our qualitative analysis, we manually typologize the counter-arguments produced by crowdworkers for CAG and CAG-F for the same 20 instances in terms of their usefulness as constructive feedback with the following: *useful*, *useless*, and *undecided*.

Our results are shown in Table 2. We observe that roughly 19% of instances were considered useful for CAG. On the other hand, with CAG-F, we were able to acquire more useful instances. In the case of CAG-F, we found that the counter-arguments were useful and specific to the fallacy type to identify. For useless examples for CAG, we found that roughly 39% of the counter-arguments were a simple contradiction (e.g., adding "not" to a claim or premise), whereas

Claim	Premise	CAG	CAG-F
People are getting dumber	In highly respected publications there are numerous errors.	*All great people sometimes make mistakes, this is not a sign of a lack of intelligence.*; *Not every highly respected publications there are numerous errors.*	*Because there are errors in publications does not mean that people are becoming more dumber*; *It can not be assumed that people are becoming more foolish because of errors in publications*
Unpaid internship exploit college students	Interns are replacing employees	*Interns should not replace employees but employees should teach interns what is needed to be done while in that position.*	*The premise does not argue for unpaid internships.*

Table 3: Examples of Useful Counter-Arguments for CAG and CAG-F

in CAG-F, we found many of the useless instances were erroneous responses from workers (e.g., counter-arguments that only said "objection" or "off-topic"). We assume that adding a verification phase similar to CAG will eliminate such erroneous candidates.

Table 3 shows examples of counter-arguments labeled as *useful* for CAG and CAG-F. While we observe that some counter-arguments produced by crowd-workers without fallacy identification are useful (e.g., *Not every highly respected publications there are numerous errors.* in Table 3), we observe that counter-arguments for CAG-F are relevant to a specified fallacy type. For example, for CAG-F, the first row indicates two counter-arguments for a *Questionable Cause* fallacy and the second row indicates a counter-argument for a *Red Herring* fallacy.

4 Conclusion and future work

Towards automatically generating constructive feedback, in this work, we experimented with constructing a corpus of user-generated counter-arguments via crowdsourcing. We conducted two parallel crowdsourcing tasks where, given a topic, claim and premise, workers were instructed to i) produce a counter-argument, and ii) first identify a fallacy type and then produce a counter-argument. Our results indicate that although we can collect counter-arguments useful as constructive feedback in both settings, especially when workers were instructed to first identify a fallacy in the original argument.

In our future work, we will extend our annotation for constructing a large-scale corpus of fallacy type and user-generated counter-arguments. First, we will expand upon the number of fallacy types to identify. Next, we will introduce a verification stage via crowdsourcing that allows workers to typologize the user-generated counter-arguments as either useful and not useful. This will allow us to collect useful instances at a large scale. After we obtain a reasonable amount of counter-arguments, we will implement a sequence-to-sequence model for automatically generating constructive feedback for a given argument.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 15H01702 and JST CREST Grant Number JPMJCR1513, including AIP challenge.

References

[1] Bo Bennett. *Logically fallacious: the ultimate collection of over 300 logical Fallacies (academic edition)*. eBookIt. com, 2012.

[2] Niamika El Khoiri and Utami Widiati. Logical fallacies in indonesian efl learners' argumentative writing: Students' perspectives. *Dinamika Ilmu*, 17(1):71–81, 2017.

[3] Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of ACL (Volume 2: Short Papers)*, pages 549–554, 2016.

[4] Ivan Habernal, Patrick Pauli, and Iryna Gurevych. Adapting Serious Game for Fallacious Argumentation to German: Pitfalls, Insights, and Best Practices. In *Proceedings of the Eleventh International Conference on LREC*, 2018.

[5] Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of NAACL: HLT, Volume 1 (Long Papers)*, pages 1930–1940. Association for Computational Linguistics, 2018.

[6] Xinyu Hua and Lu Wang. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of ACL (Volume 1: Long Papers)*, pages 219–230, 2018.

[7] Rohmani Nur Indah and Agung Wiranata Kusuma. Fallacies in english department students claims: A rhetorical analysis of critical thinking. *Jurnal Pendidikan Humaniora*, 3(4):295–304, 2015.

[8] Huy V Nguyen and Diane J Litman. Argument mining for improving the automated scoring of persuasive essays. 2018.

[9] Witri Oktavia, Anas Yasin, et al. An analysis of studentsargumentative elements and fallacies in studentsdiscussion essays. *English Language Teaching*, 2(3), 2014.

[10] Isaac Persing, Alan Davis, and Vincent Ng. Modeling organization in student essays. In *Proceedings of the 2010 Conference on EMNLP*, pages 229–239. Association for Computational Linguistics, 2010.

[11] Isaac Persing and Vincent Ng. Modeling thesis clarity in student essays. In *Proceedings of the 51st Annual Meeting of ACL(Volume 1: Long Papers)*, volume 1, pages 260–269, 2013.

[12] Isaac Persing and Vincent Ng. Modeling stance in student essays. In *Proceedings of the 54th Annual Meeting of ACL (Volume 1: Long Papers)*, volume 1, pages 2174–2184, 2016.

[13] Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein. Using Argument Mining to Assess the Argumentation Quality of Essays. In *Proceedings of the 26th International Conference on COLING*, pages 1680–1692, 2016.

[14] Henning Wachsmuth, Shahbaz Syed, and Benno Stein. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of ACL (Volume 1: Long Papers)*, volume 1, pages 241–251, 2018.