CritiqueKit: A Mixed-Initiative, Real-Time Interface For Improving Feedback

C. Ailie Fraser[1], **Tricia J. Ngoon**[1], **Ariel S. Weingarten**[1], **Mira Dontcheva**[2], **Scott Klemmer**[1]

[1]Design Lab, UC San Diego
La Jolla, CA, 92093
{cafraser, tngoon, aweingar, srk}@ucsd.edu

[2]Adobe Research
Seattle, WA, 98103
mirad@adobe.com

ABSTRACT

We present CritiqueKit, a mixed-initiative machine-learning system that helps students give better feedback to peers by *reusing* prior feedback, *reducing* it to be useful in a general context, and *retraining* the system about what is useful in real time. CritiqueKit exploits the fact that novices often make similar errors, leading reviewers to reuse the same feedback on many different submissions. It takes advantage of all prior feedback, and classifies feedback as the reviewer types it. CritiqueKit continually updates the corpus of feedback with new comments that are added, and it guides reviewers to improve their feedback, and thus the entire corpus, over time.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous.

Author Keywords

feedback; critique; real-time classification; educational technology

INTRODUCTION

Feedback is a formative part of learning, especially in creative domains [3, 8, 9]. However, good feedback is rare because it is time-consuming to give and people are not consistently skilled at providing it [5, 11]. As classes grow larger and online learning becomes more prevalent, peer feedback is a common substitute for instructor feedback. However, the utility of peer feedback is often limited, because students are not explicitly taught how to provide helpful, actionable feedback [5].

What if a system could take advantage of all the feedback people have given in the past to guide reviewers toward improving their own feedback? We introduce an approach for *reusing* feedback, *reducing* it to be useful in a general context, and *retraining* the system about what is useful in real time. CritiqueKit recommends prior feedback to student reviewers and automatically analyzes their feedback in real time to help them improve it (Figure 1).

Reviewers can view and reuse previously supplied comments, which provide awareness of what other reviewers have said before and may spark ideas they would not have otherwise considered [2, 5]. To make reviewers aware of common issues, CritiqueKit sorts recommendations by usage frequency. As the reviewer types, four checkboxes update in real-time (Figure 2b) to indicate whether the comment satisfies the characteristics of high-quality feedback. CritiqueKit extends prior work by providing ambient feedback in real time, rather than prompting the reviewer to improve after they submit a comment [6, 7].

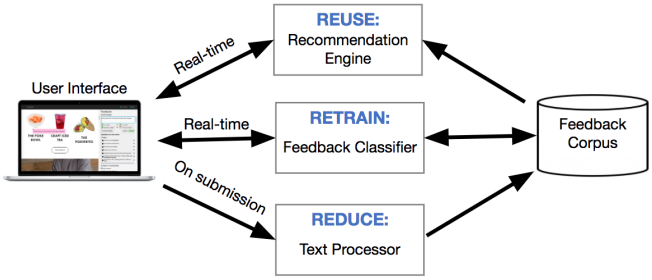

Figure 1: The CritiqueKit system components. Recommendations are generated from the feedback corpus. A machine-learning classifier categorizes feedback in real time. A text processor abstracts the user's feedback and adds it to the feedback corpus for reuse.

In an initial deployment, 20 design students used CritiqueKit to give feedback on website designs. CritiqueKit successfully engaged most students in correcting the classifier, demonstrating awareness of feedback characteristics. However, few suggestions were reused, emphasizing the need for more high-quality and relevant suggestions. Our findings suggest directions for future work to improve on these techniques.

CRITIQUEKIT SYSTEM DESIGN AND IMPLEMENTATION

The CritiqueKit architecture comprises five components: a web interface, a corpus of feedback, a feedback classifier, a text processor, and a recommendation engine (Figure 1). The system is a client-server web application implemented using a combination of Node.js and Python. The feedback corpus is stored on the server in JSON format.

CritiqueKit interface

The CritiqueKit interface allows reviewers to provide feedback on others' assignments (Figure 2). The reviewer can type their own feedback or insert previously given feedback from the list of suggested examples. As the reviewer types, four checkboxes in the quality awareness pane (Figure 2b) update as indicators of the quality of the reviewer's feedback. The reviewer can make manual corrections to the indicators, improving the system's interpretation of feedback. To enable

UIST '17 Adjunct October 22–25, 2017, Quebec City, QC, Canada

ACM ISBN 978-1-4503-5419-6/17/10.
DOI: https://doi.org/10.1145/3131785.3131791

targeted feedback, the reviewer can also attach their comment to a specific location on the document being reviewed.

Feedback corpus

The feedback corpus can be any set of feedback previously given on course assignments. The text processor and classifier automatically prepare this feedback for reuse. This serves as the initial set of suggestions that are shown in the interface. The corpus updates over time as more feedback is added.

Feedback classifier

As the reviewer types their feedback in CritiqueKit, it is dynamically classified in real-time under four different categories based on common attributes of good feedback [4, 10]: "positive", "specific", "problem", and "solution".

The "positive", "problem", and "solution" categories are informed by classifiers, which use natural language processing (primarily tf-idf) and logistic regression. Our supervised learning approach is similar to [10]: identify lexical and syntactic features from the text in the comments and train a classifier to categorize each comment. Categories are not mutually exclusive (e.g,"I like the color of the start button but the font is absolutely atrocious, try using Gotham" fits all three categories). We trained a separate classifier for each category. The training set consisted of ~5000 sentences of feedback from three undergraduate courses. Each sentence was manually categorized as positive, problem, and/or solution. CritiqueKit's classifiers outperform a naive classifier that always predicts the most common answer with F-Scores at least 0.15 higher.

The "specific" category is not informed by a classifier: a comment is labeled as specific if it is positive and at least 4 words, to discourage bland positive comments like "good job".

CritiqueKit lets reviewers correct any classification errors. These corrections supply additional labeled training examples

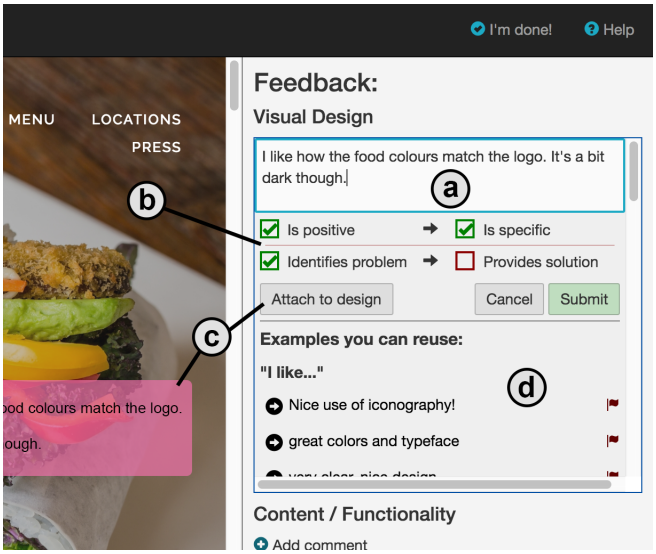

Figure 2: The CritiqueKit user interface. a) The reviewer types their feedback into the text box. b) Checkboxes in the quality awareness pane update in real-time showing how well the comment fulfills high-quality feedback criteria. c) The reviewer can attach their feedback to a specific location on the design. d) The reviewer can browse and reuse suggested feedback, and flag low-quality feedback.

of false positives and false negatives that can be used to retrain the classifiers.

Text processing of feedback for reuse

When the reviewer submits feedback, the text processor cleans it up to make it more generally applicable. It is then added to the feedback corpus, so future reviewers can reuse it from the set of examples. Based on a manual process used in initial pilot studies, the text processor excludes comments too similar to existing ones, truncates lengthy comments to 25 words or fewer, and replaces phrases in "quotation marks" or following an "e.g." with fillable blanks, as these tend to reference specific features in the original document.

Recommendation engine

The recommendation engine ranks the feedback from the corpus and presents it to the user by category (positive, problem, solution). Inspired by systems like Gradescope [1], suggestions are ordered by frequency of reuse to highlight common issues across student work. This could be improved by comparing the document being reviewed to the comment's original document, to surface the most likely relevant comments.

STUDY

We conducted an exploratory online deployment with 20 design students at a university. After viewing a brief tutorial of CritiqueKit, participants were tasked with providing feedback for three restaurant website homepages. There were no requirements for the quantity or length of feedback reviewers could provide. For our initial feedback corpus, we used a dataset of feedback given by crowd workers on design projects [11].

Results

The 20 participants provided a total of 155 comments (7 per person, SD=4.83). CritiqueKit categorized their comments as mostly problem + solution (33.55%), positive + specific (28.39%), and problem-only (27.74%) compared to positive-only (4.17%) and solution-only (10.97%). While most positive comments were specific, a significant portion stated a problem without suggesting a solution; more intervention may be needed to encourage reviewers to give actionable feedback.

67% of the reviewers actively engaged with the classifier by making corrections a total of 85 times. This indicates that most reviewers were attentive to the accuracy of their feedback classifications. However, only 23 suggestions were reused in total. Most suggestions were not directly relevant to the task and some may have been too vague to be applicable.

FUTURE WORK

We are improving CritiqueKit's classifier and refine the categories to ensure they capture high-quality feedback. We also plan to explore more effective ways to use prior feedback for inspiration, as students seemed reluctant to directly reuse comments. For example, ensuring suggestions are contextually relevant may make them more useful to reviewers. With further research, systems like CritiqueKit can help reviewers learn from each other and improve their feedback over time.

REFERENCES

1. 2017. Gradescope. (2017). `https://gradescope.com`
2. Michael D. Greenberg, Matthew W. Easterday, and Elizabeth M. Gerber. 2015. Critiki: A Scaffolded Approach to Gathering Design Feedback from Paid Crowdworkers. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition - C&C '15*. 235–244.
3. J. Hattie and H. Timperley. 2007. The Power of Feedback. *Review of Educational Research* 77, 1 (mar 2007), 81–112.
4. David Kelley and Tom Kelley. 2013. *Creative confidence: Unleashing the creative potential within us all*. Crown Publishing Group, New York. 288 pages.
5. Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R. Klemmer. 2013. Peer and self assessment in massive online classes. *ACM Transactions on Computer-Human Interaction* 20, 6 (Dec 2013), 1–31.
6. Chinmay E. Kulkarni, Michael S. Bernstein, and Scott R. Klemmer. 2015. PeerStudio: Rapid Peer Feedback Emphasizes Revision and Improves Performance. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale - L@S '15*. 75–84.
7. Huy Nguyen, Wenting Xiong, and Diane Litman. 2016. Instant Feedback for Increasing the Presence of Solutions in Peer Reviews. *HLT-NAACL Demos* (2016), 6–10.
8. D. Royce Sadler. 1989. Formative assessment and the design of instructional systems. *Instructional Science* 18, 2 (Jun 1989), 119–144.
9. Donald A. Schön. 1985. *The design studio: An exploration of its traditions and potentials*. RIBA Publications for RIBA Building Industry Trust. 99 pages.
10. W Xiong, D. Litman, and C. Schunn. 2012. Natural Language Processing techniques for researching and improving peer feedback. *Journal of Writing Research* 4, 2 (Nov 2012), 155–176.
11. Alvin Yuan, Kurt Luther, Markus Krause, Sophie Isabel Vennix, Steven P Dow, and Björn Hartmann. 2016. Almost an Expert: The Effects of Rubrics and Expertise on Perceived Value of Crowdsourced Design Critiques. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW '16*. 1003–1015.