Using Feedback Tags and Sentiment Analysis to Generate Sharable Learning Resources

Investigating Automated Sentiment Analysis of Feedback Tags in a Programming Course

Stephen Cummins, Liz Burd, Andrew Hatch
School of Engineering and Computing Sciences
Durham University
Durham, UK
s.a.cummins@durham.ac.uk, liz.burd@durham.ac.uk, andrew.hatch@durham.ac.uk

Abstract

This paper demonstrates how sentiment analysis can be used to identify differences in how students and staff perceive the opinions contained in feedback for programming work. The feedback considered in this paper is conceptually different in that it is given in the form of tags that when associated with a fragment of source code can be considered as a sharable learning resource. The research presented investigates the differences in perception of whether feedback is positive, negative or neutral according to students and examiners. This paper also investigates the adequacy of an automated sentiment analysis engine with a view that sentiment information when combined with the feedback tag and source code may create a more informative sharable learning resource. This paper describes the investigatory technique and presents the initial results. Results indicate that there are important differences between the sentiment of feedback perceived by students and examiners. This paper highlights the benefit of including sentiment data along with feedback.

Keywords; Sentiment Analysis, Feedback, Programming, Tagging

I. Introduction

Learning computer programming is recognized as a challenging undertaking throughout Higher Education [1]. Literature within the domain of educational research often cites good feedback as being critical to the success of learning and teaching [2]. It is, after all, the primary tool students have for identifying positive and negative aspects of their work. As a result improving the techniques for generation, dissemination and analysis of feedback is important to supporting both students and course directors.

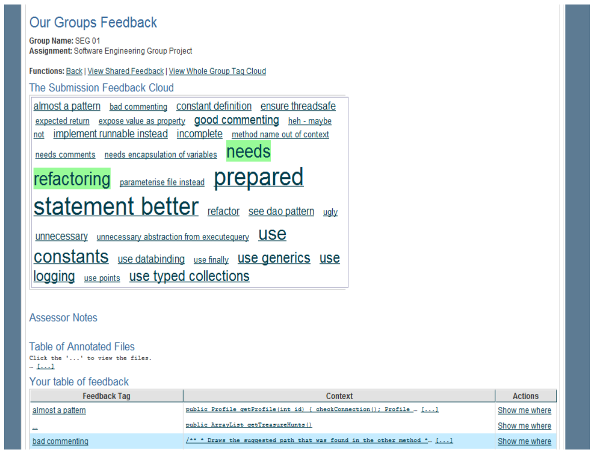
Part of the difficulty in using written feedback is that examiners and students can perceive it differently. Should the student interpret their feedback, which an examiner intended to be neutral overall, as being negative they may fail to engage with the feedback, thus rendering it less useful. It has been shown that entirely negative feedback can in fact cause students to perform worse in future assessments [3].

As in every academic discipline there are subject specific vocabularies to which some students will not be familiar with. This is especially true in a subject such as computer programming where there is a significant amount of specialized terminology. This means that students can fail to understand the messages that examiners are trying to convey.

This paper discusses a new approach for giving feedback to students programming work using tags. It also presents the results of an investigation into how positive, negative or neutral feedback tags are perceived by different participants. A comparison is made of the responses between students, examiners and an automated tool. The study focuses on investigating how automated sentiment analysis can be used with feedback tags from a programming assignment to generate useful learning resources. This supplementary sentiment data may provide additional information for students and analysis capability for examiners. A benefit to augmenting written feedback with a description of to what degree it is positive, negative or neutral makes the feedback less ambiguous and therefore mitigates one of the key problems of non-verbal feedback.

II. SWATT: SoftWare Assessment Through Tagging System

The SoftWare Assessment Through Tagging (SWATT) approach [4] focuses on using tags as a mechanism for providing feedback in the form of annotations throughout student programming work. These tag-source code associations are considered as being sharable learning resources, helping individual students to identify different approaches to solving programming problems.



Our Groups Feedback

Group Name: SEG 01
Assignment: Software Engineering Group Project

Functions: Back | View Shared Feedback | View Whole Group Tag Cloud

The Submission Feedback Cloud

almost a pattern bad commenting constant definition ensure threadsafe expected return expose value as property good commenting heh - maybe not implement runnable instead incomplete method name out of context needs comments needs encapsulation of variables needs refactoring parameterise file instead prepared statement better refactor see dao pattern ugly unnecessary unnecessary abstraction from executequery use constants use databinding use finally use generics use logging use points use typed collections

Assessor Notes

Table of Annotated Files
Click the '...' to view the files.
... [...]

Your table of feedback

Feedback Tag	Context	Actions
almost a pattern	`public Profile getProfile(int id) { checkConnection(); Profile ...` [...]	Show me where
...	`public ArrayList getTreasureHunts()`	Show me where
bad commenting	`/** * Draws the suggested path that was found in the other method *...` [...]	Show me where

Figure 1. Screenshot of the SWATT Feedback Summary View

Tagging systems facilitate information classification through the use of user generated metadata. These types of systems have been popularized by Web 2.0 platforms such as Flickr.com (photograph tagging) and del.icio.us (website bookmark tagging).

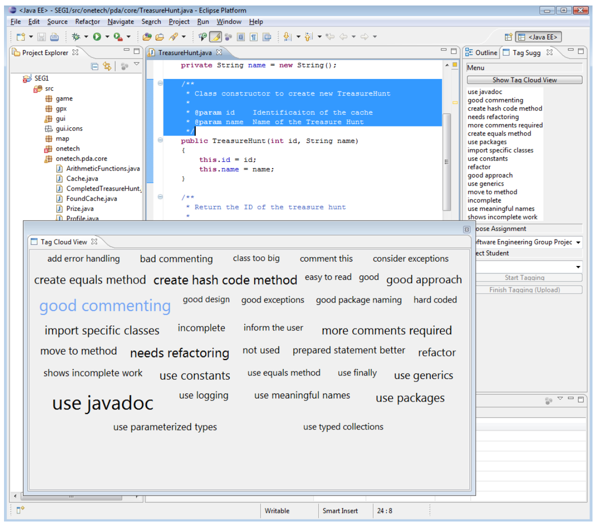
The SWATT system aims to use this tagging strategy, which has become a popular internet phenomenon as a means of encouraging students to engage with their feedback and share it. The aim is also to make the process of feedback engagement a social activity, where individuals can learn from each others achievements or mistakes in an anonymous environment.

A reason for electing to investigate the use of tags in this way is to determine whether they can be used to deliver succinct yet contextually rich feedback in an online social environment.

A. *SWATT Prototype System Architecture*

In order to help facilitate investigation into the SWATT approach for generating feedback, a prototype system was developed [4]. This system enables source code annotation using feedback tags, as well as providing functionality to support delivery and analysis of feedback for students and examiners.

The SWATT prototype is a web service developed in PHP using the CakePHP framework. The CakePHP framework was used to support rapid development of the prototype software. The web service handles the majority of the SWATT functionality including delivery of feedback to students, visualization of feedback as tag clouds, analysis of feedback, facilitating student discussion on feedback tags and handling privacy / sharing functionality for each student's feedback.



```
    private String name = new String();

    /**
     * Class constructor to create new TreasureHunt
     *
     * @param id    Identificaiton of the cache
     * @param name  Name of the Treasure Hunt
     */
    public TreasureHunt(int id, String name)
    {
        this.id = id;
        this.name = name;
    }

    /**
     * Return the ID of the treasure hunt
     *
```

Figure 2. Eclipse Tagging Plugin, showing the feedback tagging interface

In order to best support examiners it was decided to develop an interface to facilitate source code tagging in an environment that is familiar to them. As a result, a plugin for the Eclipse Integrated Development Environment (IDE), shown in Fig. 2 was developed. Eclipse was chosen since it was the IDE that students and examiners regularly used throughout the course. The plugin for Eclipse handles the recording of tag based annotations within students programming work. At the end of the tagging process the examiner uploads all of the feedback tag data to the SWATT web service which is later distributed to students who can then decide whether or not they wish to share it. The procedure for assessment using the SWATT system is as follows:

1. Student submits their assignment source code to the SWATT system.
2. Examiner uses the Eclipse plugin to annotate the students' source code with feedback tags. These are uploaded and released to the students.
3. The student can view their feedback and opt to share it. Their tags can be displayed in a cloud, as in Fig. 1 or in line with their submitted code.
4. Students that commit to the anonymous sharing feature of the SWATT system are given access to all shared feedback tags and source code.

Using a familiar IDE as a platform for the plugin means that examiners who engage in the SWATT tagging process have very little extra work to do, as it is assumed that they would be reviewing the students' submitted code using the IDE anyway. Therefore, by tagging as they go through students work, the generation feedback can be time efficient.

Whilst an eclipse plugin was selected for this particular investigation the SWATT system is not limited to one method of receiving tag data. Any system that can post eXtensible Markup Language (XML), subject to authentication, can submit tagging data.

B. *Benefits of Using Tag Based Feedback*

A consequence of using a tag based system of annotation is that the techniques commonly found in Web 2.0 environments such as tag clouds and co-occurrence analysis becomes available for the feedback. An informative method of displaying a summary of the students' feedback tags is in the form of a tag cloud. This tag cloud instantly shows the most frequently occurring tags in a students work. An example of a feedback tag cloud is shown in Fig. 1.

The notion of sharing is a core to Web 2.0 systems and as more users contribute or share information the system becomes increasingly useful. This is also true of the SWATT system. The additional information from shared work and feedback may support each student in better understanding not only their own feedback but that of their peers. For example, if a student received a feedback tag in one area of their source code and wanted clarification on its meaning; they could opt into the sharing feature of the system and be given access to view the other occasions where that tag has been used in the feedback of their peers. This additional information puts the feedback each student receives into a larger more comprehensive context.

Since programming is both a logical and creative process each student may use a variety of design and implementation approaches to develop different programs with the same output. Allowing these different approaches to be shared along with examiner feedback can support learning and can help inform students of the advantages and disadvantages of different programming practices.

A study [4] has investigated the how students engage with the sharing functionality of the SWATT system and some of the effects of using sharable tags as a form of feedback for a group programming assessment. The remainder of this paper focuses on how sentiment analysis can be used along with this novel feedback technique.

III. Sentiment Analysis

Sentiment analysis is a technique where the underlying opinion of text can be mined and examined. It allows the analyst to determine whether a particular textual input is positive, negative or neutral. The sentiment of an extract of textual data is defined in this paper as a measure of how positive, negative or neutral the underlying phraseology appears to a human reader. Typical simple examples include; "good work" which implies a positive sentiment and "bad work" a negative sentiment.

Sentiment analysis is not a new field, although, recently it has attracted a burst of research interest and continues to grow as a field [5]. A variety of terminology exists to describe this field of research which spans linguistics, computational sciences and social sciences. Some of these terms include; opinion mining, sentiment analysis, subjectivity analysis, review mining and occasionally automated approaches can be considered a form of affective computing [5].

The perceived sentiment of feedback is an important consideration in teaching as it determines how the recipient understands and acts on it. Within educational literature there are references to an approach to feedback called 'feedback sandwiches'. That is the idea that negative or critical feedback is best given when surrounded by positive comments [6]. Whilst feedback sandwiches are not always appropriate, it is usually accepted as good practice to balance the feedback that is given to ensure that it is not too negative.

A. Benefits of Using Sentiment Analysis with Feedback Tags

The SWATT approach to feedback leads to the generation of feedback tags which are associated with source code files or fragments. These learning resources inevitably will either show positive, negative or neutral aspects of programming, as identified in individual students' submitted work. These resources when shared by students can become invaluable learning aids for the cohort as a whole.

The SWATT systems focus on sharing has meant that it is even more important that the feedback given to each student and the underlying sentiment it conveys is accurately communicated from examiner to student. This is especially so as it may not be just the individual who sees the feedback anymore but all those in the cohort who opt into sharing. Hence, the importance of ensuring that feedback issued to students clearly communicates the aspects of good and bad programming practice.

Sentiment is invaluable in determining how students understand their own and other peoples' feedback. If each feedback tag was visualized according to its sentiment, then students would be able to clearly identify what they have done well and what needs improvement. Recording the sentiment of feedback is especially important in programming because words that have no inherent sentiment in natural language may have a meaning in a subject specific context. An example is "duplicate code". In the context of programming it has negative connotations as it can increase the burden of maintenance. However, it is possible some students may not realize this and as such would benefit from being informed that this tag is negative. Therefore, this paper argues that the sentiment of a feedback tag could be recorded as part of this notion of a sharable learning resources in order to reduce the ambiguity as to whether a feedback tag indicates a positive, negative or neutral aspect of programming work.

Recording sentiment analysis of feedback tags during their creation provides clear benefits to examiners in terms of enabling analysis strategies that would otherwise be unavailable.

An example of the information an examiner can discover using sentiment analysis includes determining the aspects of programming that causes students difficulty within a particular cohort. In order to achieve this they simply need to search for the feedback tags that have a negative sentiment and simply visualize the results as a tag cloud. The results of this search would provide them with an ability to immediately detect any common difficulties that may be affecting a cohort and address them in future lectures. Alternatively, they can also investigate the strengths of a particular cohort by searching for the positive tags. This information may also be useful to students in helping them identify their strengths and weaknesses from within their own work as well as that of their peers.

However, expecting examiners to highlight the sentiment of each feedback tag they generate would lengthen the assessment process and may be unrealistic in practice. An automated approach for analyzing the feedback tags for their underlying sentiment is needed. This means that the examiner merely needs verify that the sentiment identified automatically is correct according to what they intended at the end of the marking process.

B. Automatic Sentiment Analysis

A variety of automated sentiment analysis tools exist, one particular system is for extracting the sentiment from arbitrary sentences and was developed by the National Centre for Text Mining (NaCTeM) [7-8]. It is this tool which is investigated in this paper.

Automated sentiment analysis is a difficult activity to achieve computationally. This is primarily because interpreting human opinion which is naturally subjective is reliant on a number of factors including context, prior knowledge and even how the human is feeling at the time.

Automatic sentiment analysis of feedback is not a new concept and is used along with a number of useful metrics in the CAFEX2 project [9]. However, the difference between the SWATT approach and the CAFEX2 project is the focus on sharing tag based feedback as opposed to standard sentence based communication.

The following sections shall investigate how similar the NaCTeM automated sentiment analysis results are in comparison to human participants. These results will support evaluation of whether the automatically generated sentiment analysis data could be used along with feedback tags and source code to form a sharable learning resource.

IV. Investigation Method

In order to investigate how human participants' sentiment analysis of feedback tags compare with the NaCTeM automated sentiment analysis tool', an investigation was designed. The feedback tags generated from the assessment of a year long software engineering group project were collected. These were analyzed for their underlying sentiment in isolation from the associated source code fragments in order to evaluate the sentiment of each tag.

12 Java programming projects were assessed; of which 45 feedback tags, roughly corresponding to one third of all tags generated were used as a sample for sentiment analysis. As shown in Table I, all tags were analyzed using the National Centre for Text Mining (NaCTeM) sentiment analysis tool. The human analysis aspect was much more limited due to the time consuming nature of manual sentiment analysis. The human participants were limited to two randomly selected students (S_1 and S_2) and the two examiners (E_1 and E_2), those who were involved in the assessment process. The 45 feedback tags were split into 3 groups of 15 tags; labeled T_1, T_2 and T_3. T_1 was analyzed by all participants and T_2 and T_3 were analyzed by at least one examiner and one student.

The effects of the distribution shown in Table I was to ensure that at least the same 15 tags (T_1) were reviewed by every participant and a further 30 tags (T_2 and T_3) were reviewed by at least one examiner and one student. The primary purpose for limiting the amount of feedback tags seen by each human participant was to ensure that issues such as participant fatigue did not effect the investigation.

TABLE I. Distribution of Tags for Analysis

	S_1	S_2	E_1	E_2	NaCTeM Tool
T_1	✓	✓	✓	✓	✓
T_2	✓		✓		✓
T_3		✓		✓	✓

V. Results and Evaluation

The results in this study show that there is a high level of agreement between the participants and the NaCTeM tool. After comparing the 45 different tags an average of 88% agreement was reached between examiners, students and the NaCTeM tool. This percentage is calculated from the average percentage agreement for each tag.

64% of tags had complete agreement between all respondents including the automated system. The disagreements on the tags usually came from only one of the respondents (or the NaCTeM tool) and not all respondents disagreeing with each other. Table II shows the percentage tags distribution according to each respondent.

A total of 16 (36%) tags had some disagreement from students, examiners or the NaCTeM tool. Of these 16 disagreements, 6 (37.5% of all disagreements) were due entirely from the automated system disagreeing with all of the human participants. One example of these tags is the tag "ensure threadsafe" this was marked as neutral by all human respondents but the NaCTeM tool reported it as positive. These disagreements may occur due to the NaCTeM tool being unable to properly interpret computer programming specific vocabulary. It is also important to note that the NaCTeM tool is not specifically designed for short inputs such as tags and may be more effective with longer comments. Out of the remaining 10 disagreements, 6 were students disagreeing with the examiners and the NaCTeM tool. Most of the disagreements from students were suggesting that a particular feedback tag was positive or negative where as both examiners and the automated tool reported these tags as being neutral.

The final disagreements were where the students and the NaCTeM tool were in disagreement with the examiners. In one of these the students and tool recorded a sentiment of negative and the examiners thought it was neutral. The tag was "needs comments".

The results presented indicate that in this particular investigation the examiners seemed to classify fewer tags as being positive compared to the students and the NaCTeM tool. These results indicate the examiners are more likely to identify neutral and negative tags. Conversely, the students seemed to identify more positive tags, but were also more likely to perceive tags that the system and examiners noted as being neutral as being negative.

T_1 represents the tags that were reviewed by all human participants; and consequently is the most representative data set. It is interesting that this dataset confirms the trend that students identify 3% more tags as being positive and 10% more tags being negative, where as examiners consider a majority of the feedback as being neutral.

Based on the results presented in Table II, the NaCTeM tool, has achieved a mean agreement that is within 10% of the averages of the human participants. This, however, only indicates that a similar proportion of tags have been associated with the sentiments. The 64% agreement is indicative that the system is unable to cope with domain specific terminology. As a result the tool can be deemed as being useful for providing an indication of how humans may perceive feedback tags. This could be useful for automatically generating sentiment data to go along with feedback but the examiner is still required for verification.

VI. Threats To Validity

The primary threat to validity in this study is that the number of student and examiner participants involved in the sentiment analysis is small; this is due to the time consuming nature of manual sentiment analysis. Due to the small sample, the results presented may not be applicable outside the context of this investigation without further research.

Due to the nature of how the tag sample was split, some tags were not seen by every human participant. This was due to time constraints, and the study could be improved by requiring each participant to review every tag.

VII. Conclusion

This study has found that the NaCTeM tool does not always match human perception but it does provide an indication of how another person may perceive feedback tags. The information generated can be used by examiners as a guide to ensure the feedback issued matches their intended sentiment.

The NaCTeM tool can be used as a convenient method of generating the sentiment data from feedback tags as they are created and used. This information if provided to students along with the feedback could help ensure the feedback they receive is less ambiguous and more useful as a sharable learning resource.

The results of this study highlight the NaCTeM tools inability to intelligently generate sentiment information for programming specific terminology. This means that examiners will have to manually verify the sentiment of the feedback they generate before automatically including it with the feedback to students.

It is important to note that for the feedback tags considered in this study it has been shown that students and examiners do not necessarily always perceive feedback on programming work in the same way. For example, examiners may generate feedback that they perceive to be neutral but is actually seen as being negative by students. This is concerning as differences in perception can cause students to become disengaged with their feedback and negatively effect future performance [3]. This highlights a reason why feedback in tag form and perhaps in full sentences, could benefit from being associated with information that clarifies the examiners intended sentiment.

For evaluating the use of tags as a feedback mechanism, this investigation was limited because the feedback tags were considered in isolation from the associated source code. This was to keep consistency between human and automated respondents in the study. This means that the context that would normally be given from where the tag was associated was hidden. Given this restriction, the responses for feedback tags were remarkably consistent between examiners, students and the NaCTeM tool, with almost a ⅔ complete agreement.

In conclusion, automated sentiment analysis tools can be useful both in automatically generating associated sentiment information for feedback to be communicated to students but also for providing an indication of how feedback may be perceived by a student if the sentiment data is not explicitly provided.

VIII. Future Work

Based on these results, it has been decided that in future versions of the SWATT system the sentiment of feedback tags could be visualized in the tag cloud using color to represent the different types. This information could help communicate the intended sentiment to students visually, and perhaps reduce the ambiguity of the feedback tags. It is however, noted that feedback given in other forms may also be improved by including this sentiment data. Before sentiment analysis is fully incorporated into the SWATT system further research is required on how students would respond to this type of feedback and whether they would share the sentiment information with their peers.

Future research could be carried out to investigate improvements to automated sentiment analysis tools in interpreting programming specific feedback. This would provide examiners with less of a need to verify the sentiment generated by automated systems.

The investigation presented is part of an ongoing investigation into tag based feedback for programming assessment. Future research shall investigate how students utilize this type of feedback and ultimately whether or not there is a perceived benefit to the overall approach.

References

[1] Robins, A., Rountree, J. and Rountree, N. "Learning and Teaching Programming: A Review and Discussion", Computer Science Education, 13(2), 2003, pp. 137-172.

[2] Laurillard, D. "Rethinking University Teaching: a framework for the effective use of educational technology". Routledge, London, 1993.

[3] Gee, T. C. "Students Responses to Teacher Comments" , Research in the Teaching of English, 6(2), 1972, pp. 212 - 221.

[4] Cummins, S., Burd, L. and Hatch, A. "Tag Based Feedback for Programming Courses", ACM SIGCSE Bulletin (inroads), 41(4), December 2009, pp 62-65

[5] Pang, B. and Lee, L. "Opinion Mining and Sentiment Analysis", Foundations and Trends in Informational Retrieval 2(1-2), 2008, pp. 1-135

[6] Haines, C. "Assessing students' written work: marking essays and reports", Routledge, 2004.

[7] Piao, S., Tsuruoka, Y. and Ananiadou, S. "Sentiment Analysis with Knowledge Resource and NLP Tools", International Journal of Interdisciplinary Social Sciences, 4(5), 2009, pp.17-28

[8] The National Centre for Text Mining (NaCTeM), http://www.nactem.ac.uk/software.php, Accessed 25/08/2009

[9] Gillam, L., Qin, G., Bush, D. and Newbold, N. "Automating Feedback: The CAFEX2 Project", The Higher Education Academy, Subject Centre for Information and Computer Sciences 10th Annual Conference, University of Kent at Canterbury, August 2009

TABLE II. Results of Sentiment Analysis for Tag Samples T_1, T_2 and T_3

Sentiment	**Students**				**Examiners**				**NaCTeM Tool**			
	T_1	***T_2***	***T_3***	***Mean***	***T_1***	***T_2***	***T_3***	***Mean***	***T_1***	***T_2***	***T_3***	***Mean***
Positive	10%	7%	14%	10%	7%	7%	14%	9%	13%	13%	7%	11%
Neutral	63%	60%	50%	58%	77%	60%	64%	67%	73%	40%	86%	66%
Negative	27%	33%	36%	32%	17%	33%	22%	24%	13%	47%	7%	23%