Detecting Problem Statements in Peer Assessments

Yunkai Xiao [yxiao28],[1] Gabriel Zingle [gzingle],[1] Qinjin Jia [qjia3],[1] Harsh R. Shah [hshah3],[1]
Yi Zhang [20171184],[2] Tianyi Li [tianyi.li],[3] Mohsin Karovaliya [mrkarova],[1]
Weixiang Zhao [20172986],[2] Yang Song [songy],[4] Jie Ji [20172986],[5]
Ashwin Balasubramaniam [abalasu4],[1] Harshit Patel [hpatel24],[1]
Priyankha Bhalasubbramanian [pbhalas],[1] Vikram Patel [vpatel22],[1] and Edward Gehringer [efg][1]

[1] North Carolina State University, Raleigh, North Carolina 27695, USA [@ncsu.edu]
[2] Northeastern University, Shenyang, Liaoning 110819, China [@stu.neu.edu.cn]
[3] Shanghai Jiao Tong University, Shanghai 201101, China [@sjtu.edu.cn]
[4] University of North Carolina at Wilmington, Wilmington, North Carolina 28407, USA [@uncw.edu]
[5] Southern University of Science and Technology, Shenzhen, Guangdong 518055, China [@mail.sustech.edu.cn]

ABSTRACT

Effective peer assessment requires students to be attentive to the deficiencies in the work they rate. Thus, their reviews should identify problems. But what ways are there to check that they do? We attempt to automate the process of deciding whether a review comment detects a problem. We use over 18,000 review comments that were labeled by the reviewees as either detecting or not detecting a problem with the work. We deploy several traditional machine-learning models, as well as neural-network models using GloVe and BERT embeddings. We find that the best performer is the Hierarchical Attention Network classifier, followed by the Bidirectional Gated Recurrent Units (GRU) Attention and Capsule model with scores of %93.1 and %90.5 respectively. The best non-neural network model was the support vector machine with a score of 89.71%. This is followed by the Stochastic Gradient Descent model and the Logistic Regression model with 89.70% and 88.98%.

Keywords

Peer assessment, problem detection, text mining, text analytics, machine learning

1. INTRODUCTION

Peer assessment—students giving feedback on each other's work—has been a common educational practice for at least 50 years [1, 2] It provides students more copious and rapid feedback than an instructor would give, as well as reactions from a more authentic audience (the student's peers). By concentrating on a limited number of works, peers can produce assessments with similar validity and reliability to those of instructors, whose time is spread more thinly over many students' submissions [3]. Students who perform peer assessment show a substantial increase in performance [4]. Moreover, studies uniformly report that students learn more by being reviewers than they learn from the reviews they receive [5, 6, 7, 8].

Yunkai Xiao, Gabriel Zingle, Qinjin Jia, Harsh Shah, Yi Zhang, Tianyi Li, Mohsin Karovaliya, Weixiang Zhao, Song Yang, Jie Ji, Ashwin Balasubramaniam, Harshit Patel, Priyankha Bhalasubbramanian, Vikram Patel and Edward Gehringer "Detecting Problem Statements in Peer Assessments" In: *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, Anna N. Rafferty, Jacob Whitehill, Violetta Cavalli-Sforza, and Cristobal Romero (eds.) 2020, pp. 704 - 709

The need for peer assessment was felt more acutely after the rise of massive open online courses (MOOCs). With students paying little to no fees, MOOCs are not able to hire enough staff to asses all submitted work. Thus, MOOCs rely heavily on peer assessment [9, 10].

For students to gain from peer assessment, students must take the process seriously. They must think carefully and metacognitively about the works they are reviewing. To foster an atmosphere where students assess conscientiously, the instructor must train the students in reviewing—and follow up by assessing how well they perform this task [11]. But instructor assessment of students' reviewing suffers from the same shortcomings as instructor assessment of students' submitted work: it consumes much instructor time, is likely to be rushed, and is mostly summative; that is it evaluates how well the students have done, but does not directly help them improve their reviewing. Thus, considerable research has looked at other methods for assessing review quality [12].

Fundamentally, the quality of a review is related to whether it identifies ways for the author to improve the work. Thus, the review should point out shortcomings or problems the reviewer perceives in the reviewed work. This paper describes several approaches to automatically identifying whether review *comments*, which are responses to individual rubric items, do point out (alleged) problems with the work.

2. RELATED WORK

Previous approaches to evaluating peer-assessment reviews include calibration [13, 10, 14], reputation systems [15, 16], "back-reviews" (rejoinders) [17], natural language processing [18, 19, 20], logistic regression [21], and neural-network techniques [22]. Peer assessment has much in common with peer review, as used to vet scientific work for publication. Hua et al. [23] used NLP to automatically detect arguments in these reviews. Negi [24] used several AI techniques to detect suggestions in product reviews. Space does not permit elaboration of these methods, but a fuller discussion can be found in our extended paper.

3. EXPERIMENTAL METHODOLOGIES

3.1 Data

The data used for our experiments comes from Expertiza [25], a peer-assessment platform for reviewing work developed by collaborative teams. For each review, the reviewer fills out a rubric, which consists of several criteria. Sample rubric items are, "How well does the code follow good Ruby and Rails coding practices?" "Is the user interface intuitive and easy to use?" Most criteria ask for a numeric rating as well as textual feedback. It is the textual feedback that we analyze in this work.

Our study is based on reviews of coding and documentation assignments from NCSU CSC 517, Object-Oriented Design and Development. To obtain labeled data for our research, we offered students a small amount of extra credit for tagging review comments they received, as either mentioning a problem or not. We spot-checked the student-assigned tags for the purpose of quality control. An example comment that does not mention a problem (tagged as 0) is, "The interface is easy to use and it is well described in the Readme file." One mentioning a problem (tagged as 1) is, "The implementation can only log one type of user on."

Several students had the opportunity to tag the same review comments. If multiple students tagged the same comment, inter-rater reliability (IRR) could be calculated. We used Krippendorff's α [26] as the metric for IRR. By dropping observations with conflicting tags, we have raised the Kirppendorff's α associated with our dataset from 0.696 to 1.

The dataset was de-duplicated and balanced, resulting in a total of 18,354 observations. It was separated into training, validation, and testing sets in the ratio of 80:10:10. This split was used to find optimized hyperparameters with 5-fold cross-validation. Unless the dataset is large, the combination of observations used in the training and test sets can have an impact on how well the classifier performs. We compensated for this by using 20-fold cross-validation on our finalized classifiers with tuned hyperparameters and saving the resulting 20 scores for analysis.

3.2 Baseline Models

We set up our baseline using traditional machine-learning models, such as Support Vector Machine (SVM), SVM using Stochastic Gradient Descent (SGD), Multinomial Naïve Bayes (MNB), Logistic Regression (LR), Random Forest (RF), Gradient Boosting (GB), and AdaBoost (AB).

3.2.1 Input Embedding

The input to our baseline models was first processed by the TF-IDF vectorizer in scikit-learn [27]. TF-IDF vectorization is a common way to convert raw text and documents into embeddings suitable for machine-learning models. The vectorizer generates a document-vocabulary matrix for each of the documents (in our case, review comments that averaged 2.2 sentences per comment). Then, using inverse document frequency, it normalizes ("lowers") the weight of the words by checking how often a word appears in other documents (comments, in this case). This helps lessen the impact of frequent yet unimportant words, so that common words like "the" that convey little semantic meaning do not affect the classification of a comment. The model architecture and dataflow for traditional classifiers is shown in Figure 1.

3.2.2 Support Vector Machine

Support vector machines are commonly used for classification in machine learning. A SVM establishes a decision boundary as well as a positive plane and a negative plane between classes. Statistical features for each review comment represented in TF-IDF-normalized vectors are put into the vector space for all comments, then the model learns a hyper-plane (support vector) to best divide them into two categories: comments containing problem statements, and comments without problem statements.

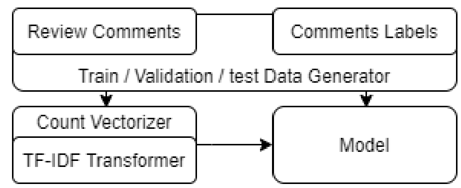

Figure 1: Data pipeline for machine learning model

3.2.3 SVM with Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) was developed early on and popularly adopted to optimize neural-network models [28], while applying SGD on linear classifiers is not unheard of. [29] We compared the performance of the SVM model with and without SGD.We applied a combination of L1 and L2 regularization to the loss function, with the hope of correcting over-fitting problems.

3.2.4 Multinomial Naive Bayes

A naïve Bayes model assumes that each of the features it uses for classification is independent the others. To determine whether a review comment identifies a problem, the model examines the TF-IDF normalized word-count vectors for that comment, using the conditional probability of each of these features/vectors, and makes a judgment, based on conditional probabilities learned from the training set.

3.2.5 Logistic Regression

The logistic-regression (LR) classifier uses a regression equation to produce discrete binary outputs. Similar to linear regression, it learns the coefficients of each input feature through training; however it uses a logistic function instead of linear activation to determine the class to which an input belongs by fitting coefficients of each n-gram through comments in the given training set.

3.2.6 Random Forest

The Random Forest (RF) classifier is an ensemble method that fits multiple decision trees and uses averaging to improve the accuracy of predictions and to avoid over-fitting.

3.2.7 Gradient Boosting

Gradient boosting (GB) is an ensemble machine-learning algorithm that utilizes a number of weak models, such as small decision trees. In training, these small decision trees are fitted in a negative gradient direction in order to reduce the loss calculated from the cost function.

3.2.8 AdaBoost

AdaBoost, or adaptive boosting, is a meta-algorithm that alters weights of entries for base models. When an entry is misclassified, the algorithm increases the weight of that entry and decreases the weights of entries that have been correctly classified. The algorithm terminates upon meeting the confidence threshold. Through doing this, the booster identifies the features that have greater impact on the results, and improves prediction accuracy.

3.3 Neural Network Models

Our other experiments use neural networks, and Keras [30] was the framework of choice for implementation. Compared with our baseline models, the input of each model is generated in two different ways: through a GloVe embedding and BERT embedding.

3.3.1 Input Embedding

Global Vectors for Word Representation, or GloVe embedding [31], is an embedding model that converts words into multidimensional vectors based on their meaning. Its function is similar to Word2Vec, which transforms words to embeddings in a limited vector space, though the underlying principle is different.

Bidirectional Encoder Representations from Transformers (BERT) is a multi-layer bidirectional transformer encoder [32] developed by Google. The BERT network we used in our experiment is published by Google and is pre-trained on Wikipedia and BooksCorpus data. We used the open-source project ”Bert-as-service” to create sentence embeddings. Specifically, we limited the maximum sentence length to 25 words, and extracted embeddings with outputs from the second-last layer in the pretrained network. The Bert-Base-Uncased model [32] has 12 attention layers, and 768 neurons in each layer with 12 attention heads. Using this network has given us 768 dimensions as sentence embeddings. We also used a version with word level embeddings. Figure 2 demonstrates the model architectures in order of the next subsection.

3.3.2 Multilayer Perceptron

A multilayer perceptron (MLP) model [33] is a typical artificial neural network. It utilizes multiple layers of neurons, and uses back-propagation for training. Errors calculated by a loss function are propagated back through the layers using the chain rule of gradient descent derivation.

3.3.3 Convolutional Neural Network

A convolutional neural network (CNN) utilizes convolution kernels that pool data with a defined window size on given dimensions to generate summaries from input data [34].

When dealing with comment classification, this model uses convolutions on the feature dimension to reduce the complexity of each word vector, different dropout percentages, and pooling methods.

→

Layer	BERT	Input	Dense	Dense	Dense
Size	768	768	16	12	1

(a) BERT + MLP

→

Layer	GloVe	Drop out	Convolution	Batch Normalization	Global avg pooling	Dense	Drop out	Dense
Size	(100,100)	(100,100)	(100,50)	(100,50)	50	100	100	2

(B) GloVe + CNN

→

Layer	GloVe	Drop out	Bi-LSTM	Drop out	Dense
Size	(80,100)	(80,100)	(100,50)	(80,100)	2

(c) GloVe + Bi-LSTM

→

Layer	Input	Time distributed	Bi-GRU	Attention	Drop out	Dense
Size	(3,32)	(3,128)	(3,128)	128	128	2

(d) HAN

→

Layer	GloVe	Drop out	Convolution	Average Pooling	LSTM	Drop out	Dense
Size	(80,100)	(80,100)	(80,50)	(20,50)	100	100	2

(e) GloVe + CNN + LSTM

→

<table>
<tr><td rowspan="2">Layer</td><td rowspan="2">GloVe</td><td rowspan="2">Drop out</td><td rowspan="2">Bi-GRU</td><td>Capsule</td><td>Flatten</td><td>Drop Connect</td><td rowspan="2">Concatenate</td><td rowspan="2">Dense</td><td rowspan="2">Dense</td></tr>
<tr><td>(10,10)</td><td>100</td><td>32</td></tr>
<tr><td rowspan="2">Size</td><td rowspan="2">(100,300)</td><td rowspan="2">(100,300)</td><td rowspan="2">(100,600)</td><td colspan="2">Attention</td><td>Drop Connect</td><td rowspan="2">48</td><td rowspan="2">100</td><td rowspan="2">2</td></tr>
<tr><td colspan="2">600</td><td>15</td></tr>
</table>

(f) GloVe + Bi-GRU + Attention + Capsule

Figure 2: Data pipeline for neural network Models

3.3.4 Recurrent Neural Network

Recurrent neural networks (RNNs) are neural networks that take time-sequence information into consideration. For each time-step, the network takes the inputs and updates its internal memory cells with new information. Different RNN models implement memory updates differently. For example, long short-term memory (LSTM) networks not only remember inputs, but also ”forget” unimportant information.

When we pass an embedded sentence to the network, each word is seen as an item emerging in one time step, and the sequence of words in a sentence becomes a sequence of vectors transitioning along with time steps. The neural network learns from the transition what information is important to keep and what is not, then applies the same judgment when a new sentence is given to it for classification.

Here we also implemented a GRU network and a bidirectional GRU network in parallel.

3.3.5 Hierarchical Attention Network

Hierarchical attention networks (HANs) are neural networks that take into consideration the document structure and sentence structure [35]. A document normally consists of a number of sentences, and a sentence is formed by a number of words. Not all sentences in a document are important to the classification of a document, and similarly, not all words are important for sentence-level classification. HANs utilize this information through attention layers that capture words and sentences that are important towards the classification.

In classifying comments, a HAN can capture information with greater impact on the results. For example in sample comment ”The writeup does not include a Test Plan section,” the words ”does not include” contributes a lot more to implying there is a problem stated in this comment than other parts of the comment do.

3.3.6 CNN with Long Short Term Memory

Previous models showed that each type of the neural network or neural network layer could be efficient on specific tasks, for example CNN for dimension reduction and HAN for extracting words that are more important to the result. In this subsection we combine some models and explore the benefits of mixing different types of neural networks.

A model with CNN and LSTM layers is implemented in the hope of securing benefits from both models. With CNN as a dimension reducer, the LSTM layer might be able to find useful information from the aggregated features. Another attempt tests whether a CNN is needed to reduce dimensions, by removing it while boosting the performance of recurrent layer by putting it in a bidirectional wrapper.

4. EXPERIMENTAL RESULTS

Figure 3 displays a boxplot of the 20 f1-scores obtained using the traditional machine-learning classifiers and neural networks from the 20-fold cross validation. The lowest-performing classical machine learning classifiers, multinomial naïve Bayes and AdaBoost, achieved similar accuracy, with respective sample median f1-scores of 0.855 and 0.861. The gradient boosting and random-forest classifiers achieved sample median f1-scores of 0.870 and 0.871. The highest performing classifiers included logistic regression, stochastic gradient descent, and support vector machines. They achieved sample median f1-scores of 0.890, 0.897, and 0.897 respectively.

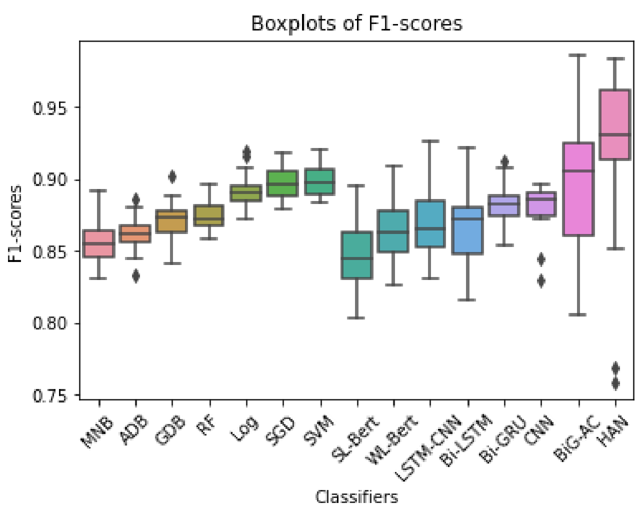

Figure 3: Models' F-1 Scores

These results show that classifiers can classify review comments as mentioning problems with an accuracy range of approximately 84% to 95%.

The HAN and BiGRU-Attn-Caps models that used GloVe embeddings achieved the best performance among all the models. The CNN model that used GloVe embeddings achieved the next best performance with a sample median f1-score of 0.886. The Bidirectional GRU had a very close sample median f1-score of 0.882, followed by the Bidirectional LSTM model with 0.872, then the LSTM CNN model at 0.865. The lowest-scoring models were the ones with word-level (WL-Bert) and sentence-level (SL-Bert) BERT embeddings with sample median f1-scores of 0.862 and 0.844 respectively.

To gain insight into the phrases that contributed towards determining a suggestion, we extract coefficient weights of some features from two of the models. Table 1 displays a list of the logistic regression model's top 10 positive and negative features in determining if a comment has mentioned a problem in the author's work. The features that increase the likelihood that a comment will mention a problem (positive coefficients) include phrases that may constitute a suggestion by the reviewer. For instance, phrases such as ”could”, ”should”, ”could have”, and ”more” indicate that the reviewer is likely giving advice to the author about improving the work, thus noting a problem by implication. Features with negative coefficients include phrases that likely demonstrate positive sentiment, such as ”yes”, ”good”, ”well”, and ”great”.

Table 1: Logistic Regression Coefficients

Coefficient	Value	Coefficient	Value
yes	-8.0233	not	10.5227
good	-3.9472	but	8.8498
and	-3.1690	however	7.8254
they have	-3.1193	more	6.2155
well	-3.0567	could	5.6703
yes the	-2.9953	should	5.3498
all the	-2.7422	would	5.0391
clearly	-2.6269	no	5.0183
project	-2.5331	missing	4.9864
passed	-2.4645	some	4.9160

Table 2 displays the stochastic gradient descent model's top 10 positive and negative features in determining if a comment mentioned a problem in the author's work. The coefficient values are lower than those of the logistic regression model, but they comprise similar positive and negative features.

Table 2: Stochastic Gradient Descent Coefficients

Coefficient	Value	Coefficient	Value
yes	-4.1029	however	6.5277
conflicts	-2.0396	not	6.4184
good	-2.0083	but	5.5175
apply	-1.7788	should	3.9721
complicated	-1.7785	could	3.9198
since	-1.6178	would	3.8352
sense	-1.6139	more	3.6346
required	-1.5925	missing	3.5942
passed	-1.5757	no	3.4112
project	-1.5637	except	2.9776

5. SUMMARY

We have marshalled a multitude of classifiers that can parse student peer-review comments for the detecting the mention of a problem. The HAN and BiGRU-Attn-Caps models performed the best among the neural network classifiers on this dataset, while the best traditional classifiers were the support vector machine and stochastic gradient descent models. The least effective classical models were the AdaBoost and multinomial naïve Bayes classifiers—the two that used the sentence and word level embeddings.

6. REFERENCES

[1] Keith Topping. Peer assessment between students in colleges and universities. *Review of Educational Research*, 68(3):249–276, 1998.

[2] Joanna Tai and Chie Adachi. 5 the transformative role of self-and peer-assessment in developing critical thinkers. *Innovative Assessment in Higher Education: A Handbook for Academic Practitioners*, 2019.

[3] Keith J Topping. Peer assessment. *Theory into Practice*, 48(1):20–27, 2009.

[4] Hongli Li, Yao Xiong, Charles Vincent Hunter, Xiuyan Guo, and Rurik Tywoniw. Does peer assessment promote student learning? a meta-analysis. *Assessment & Evaluation in Higher Education*, pages 1–19, 2019.

[5] Kristi Lundstrom and Wendy Baker. To give is better than to receive: The benefits of peer review to the reviewer's own writing. *Journal of Second Language Writing*, 18(1):30–43, 2009.

[6] Yasemin Demiraslan Çevik. Assessor or assessee? investigating the differential effects of online peer assessment roles in the development of students' problem-solving skills. *Computers in Human Behavior*, 52:250–258, 2015.

[7] Lan Li, Xiongyi Liu, and Allen L Steckelberg. Assessor or assessee: How student learning improves by giving and receiving peer feedback. *British Journal of Educational Technology*, 41(3):525–536, 2010.

[8] Esther Van Popta, Marijke Kral, Gino Camp, Rob L Martens, and P Robert-Jan Simons. Exploring the value of peer feedback in online learning for the provider. *Educational Research Review*, 20:24–34, 2017.

[9] Judy Kay, Peter Reimann, Elliot Diebold, and Bob Kummerfeld. MOOCs: So many learners, so much potential ... *IEEE Intelligent Systems*, 28(3):70–77, 2013.

[10] Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong Do, Andrew Ng, and Daphne Koller. Tuned models of peer assessment in MOOCs. *arXiv preprint arXiv:1307.2579*, 2013.

[11] Xiongyi Liu and Lan Li. Assessment training effects on student assessment skills and task performance in a technology-facilitated peer assessment. *Assessment & Evaluation in Higher Education*, 39(3):275–292, 2014.

[12] Edward F Gehringer. A survey of methods for improving review quality. In *International Conference on Web-Based Learning*, pages 92–97. Springer, 2014.

[13] Orville L Chapman and Michael A Fiore. Calibrated peer review$^{\mathrm{TM}}$. *Journal of Interactive Instruction Development*, 12(3):11–15, 2000.

[14] Yufeng Wang, Hui Fang, Qun Jin, and Jianhua Ma. SSPA: an effective semi-supervised peer assessment method for large scale MOOCs. *Interactive Learning Environments*, pages 1–19, 2019.

[15] John Hamer, Kenneth TK Ma, and Hugh HF Kwong. A method of automatic grade calibration in peer assessment. In *Proceedings of the 7th Australasian Conference on Computing education-Volume 42*, pages 67–72. Australian Computer Society, Inc., 2005.

[16] Kwangsu Cho and Christian D Schunn. Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Computers & Education*, 48(3):409–426, 2007.

[17] Luca De Alfaro and Michael Shavlovsky. Crowdgrader: A tool for crowdsourcing the evaluation of homework assignments. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 415–420. ACM, 2014.

[18] Wenting Xiong and Diane Litman. Automatically predicting peer-review helpfulness. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 502–507. Association for Computational Linguistics, 2011.

[19] Wenting Xiong, D Litmaan, and Christian Schunn. Natural language processing techniques for researching and improving peer feedback. *Journal of Writing Research*, 4(2):155–176, 2012.

[20] Caroline Brun and Caroline Hagege. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70(79.7179):5379–62, 2013.

[21] Huy Nguyen, Wenting Xiong, and Diane Litman. Instant feedback for increasing the presence of solutions in peer reviews. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 6–10, 2016.

[22] Zhongcan Xiao, Chandrasekar Rajasekar, Ferry Pramudianto, Edward Gehringer, Vishal Chittoor, and Abhinav Medhekar. Application of neural-network models to labeling educational peer reviews. In *CSEDM 2018: Educational Data Mining in Computer Science Education Workshop*, Buffalo, NY, 2018. IEDMS.

[23] Xinyu Hua, Mitko Nikolov, Nikhil Badugu, and Lu Wang. Argument mining for understanding peer reviews. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2131–2137, Minneapolis, MN, June 2019. Association for Computational Linguistics.

[24] Sapna Negi and P Buitelaar. Suggestion mining from opinionated text. *Sentiment Analysis in Social Networks*, pages 129–139, 2017.

[25] Edward F Gehringer. Expertiza: Managing feedback in collaborative learning. In *Monitoring and Assessment in Online Collaborative Environments: Emergent Computational Technologies for E-learning Support*, pages 75–96. IGI global, 2010.

[26] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage Publications, 2013.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[28] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[29] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[30] François Chollet et al. Keras: Deep learning library for theano and tensorflow. *URL: https://keras. io/k*, 7(8):T1, 2015.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[33] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[35] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.