Spring 5-20-2022

Application of Crowdsourcing and Machine Learning to Predict Sentiments in Textual Student Feedback in Large Computer Science Classes

Robert Kasumba
Washington University in St. Louis

Washington University in St. Louis

McKelvey School of Engineering

Department of Computer Science and Engineering

Thesis Examination Committee:
Marion Neumann, Chair
Chien-Ju Ho
William Yeoh

Application of Crowdsourcing and Machine Learning to Predict Sentiments in Textual Student Feedback in Large Computer Science Classes

by

Robert Kasumba

A thesis presented to the McKelvey School of Engineering of Washington University in St. Louis in partial fulfillment of the requirements for the degree of Master of Science

May 2022
Saint Louis, Missouri

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to extend special acknowledgment to my advisor, Dr. Marion Neumman for her dedicated support and mentorship as I worked on this research. I would not have been able to complete this work without her guidance and direction. I am also very grateful to the students who participated in the crowdsourcing experiment and labeled the dataset used in this work.

I would also like to thank the faculty at Washington University in St. Louis that have supported me during my masters study in the past two years.

To the team at the International Center for Child Health and Development (ICHAD) especially Dr. Fred Ssewamala - Thank you for your support which has been key to my successful completion of the masters degree.

I also acknowledge the several students as well as faculty at Washington University who reviewed this thesis and gave me valuable feedback on how to improve it.

Finally, I would like to thank my masters committee for their guidance and input in this Thesis.

Robert Kasumba

Washington University in St. Louis
May 2022

Dedicated to my family and to all those that have believed in me and supported me on my academic journey.

ABSTRACT OF THE THESIS

Application of Crowdsourcing and Machine Learning to Predict Sentiments in Textual Student Feedback in Large Computer Science Classes

by

Robert Kasumba

Master of Science in Computer Science

Washington University in St. Louis, 2022

Research Advisor: Professor Marion Neumann

With the increasing enrollment numbers into popular computer science courses, there is a need to bridge the similarly increasing feedback gap between individual students and course instructors. One way to address this challenge is for instructors to collect feedback from students in form of textual reviews or unit-of-study reflections – however, manually reading these reviews is time-consuming, and self-reported Likert scale responses are noisy. Rule-based approaches to sentiment analysis such as VADER (Valence Aware Dictionary and sEntiment Reasoner) have been used to capture the sentiments conveyed in textual feedback, they however fail to capture contextual differences as many words have different sentiments in different contexts. In this work, I investigated the use of supervised machine learning approaches and compared their performance in predicting the sentiment in student feedback collected in large computer science classes with the lexicon-based approach VADER. I found that machine learning models trained solely on student self-reported sentiment ratings were only comparable with a balanced accuracy of 73.8% versus 73% (VADER) . However, a

hybrid approach using the VADER score as a feature and training using the student self-ratings performed better than VADER alone. Using better quality labels collected through a crowdsourcing experiment led to the best machine learning model performance.

Chapter 1

Introduction

Enrollment numbers of computer science courses have been on the rise since 2006 in most US universities [1, 2, 3]. This has come with a similarly widening feedback gap between students and course instructors. Collecting student feedback in form of free-form text or Likert-style survey questions throughout the semester is one approach to bridge this feedback gap. Periodic student feedback allows course instructors to learn about course materials students struggle with, gain awareness of teams that have issues, or even identify students that fall behind in a timely manner [4, 5, 6]. Once identified, students or teams at risk can be offered tailored support to improve their experience with the course or project they are working on. Analyzing student feedback to identify areas of concern for students in specific course units or assignments helps instructors to prepare for review sessions or to improve future offerings of the course.

Whereas responses to Likert-type survey questions are easy to aggregate and analyze, they have numerous major drawbacks. The survey questions need to be carefully designed and tailored to specific contexts and the responses tend to be biased towards the positive, noisy, and thus unreliable [7, 8, 9]. Furthermore, they offer limited detail as to what causes issues or misunderstandings for instance. Textual feedback not only provides more detail but is also easy to collect since instructors can use general prompts that trigger students to report their experiences or reflections for certain course units or activities. However, the large volume of textual reviews is cumbersome to manually read through or analyze. This invites the use of text and sentiment analysis tools to analyze these reviews and filter those with negative emotions for special attention by course instructors. Sentiment Analysis (SA) refers to computational approaches that determine the sentiment polarity of a piece of text. SA

has been applied to predict the polarity of movie reviews, tweets, online product feedback [10, 11, 12]. One approach to sentiment analysis are rule-based methods that maintain static lexicons mapping the polarity generally associated with a given word or set of words to a sentiment value often between -1 and $+1$. One example of such a lexicon-based is VADER (Valence Aware Dictionary for Sentiment Reasoning) [13] which was chosen as the baseline comparison in this work. Although these methods have been demonstrated the ability to capture the sentiment in textual reviews [14], they often fail to capture the overall context of the text. Some words that are known to have a positive sentiment may have neutral sentiment in other contexts [15]. The static nature of rule-based lexicons limits their application to new and dynamic contexts such as the prediction of sentiments in different computer science courses.

Supervised Machine Learning (ML) approaches are designed to overcome these issues [16, 17, 18]. One major challenge with supervised ML is that we need ground truth labels to train the models. However for text data processing, this ground truth is especially hard to come by. It is not only time-consuming to collect through manual human annotation, but also it is subjective. In this work, I investigate the use of student self-reported labels as well as crowd-sourced labels in machine learning approaches, specifically random forest and support vector machine classifiers, to predict the sentiment in student textual feedback.

I used assignment reflections and self-reported likert scale ratings collected from several large computing courses over multiple years spanning upper-level computing and introduction to computer science courses. In addition, I ran a crowdsourcing experiment in an introduction to data science course to collect multiple labels for the same feedback text to aggregate them into lesser noisy quality labels that can be used for training and evaluation.

1.1 Research Questions

To investigate whether the use of supervised machine learning and crowd-sourced ground truth sentiment labels improve the accuracy of sentiment predictions of student feedback my work was driven by the following specific research questions:

Question 1: Do supervised machine learning approaches perform better at sentiment prediction for student textual feedback compared to rule-based approaches such as VADER?

Question 2: What features and label information is needed for supervised machine learning approaches to predict the sentiment of student textual unit-of-study reflections?

1.2 Related Work

In this area, several researchers have studied the use of tools to collect and analyze student feedback. These include analysis techniques as well as visualizations tools to help class instructors interpret the feedback. Hannah J. et. al [19] used a smartphone application to collect real-time feedback during the class and provided the instructor with a dashboard with key insights from the student feedback. They found out that the insights guided the instructors to change their teaching approach to address the feedback from the students. Kai Presler-Marshall et. al [4] employed weekly surveys to monitor software engineering students teams to identify and help struggling teams. The reviews were manually analyzed by reading the submission of the students and noted that the students felt that the self-reflections helped them to stay focused on the projects. They however did not notice a significant improvement in the final grades of the students. Investigating a similar problem, Niki Gitinabard et. al [20] developed a dashboard that aggregated students interactions logs from multiple learning management systems such as Piazza, Github, Moodle to understand how different individuals in a specific team contributed to a group project. Selected course instructors used the tool to monitor how well teams were managing the dynamics of group work and easily identified struggling teams. The tool also supported instructors to send emails to the teams that needed help by providing email templates. Maija Hujala et al [21] studied the use of the Latent Dirichlet Allocation (LDA) model [22], thematic analysis, and multilevel modeling to extract topics in large student feedback or a single course or at an institution-wide level. This approach (LDA) was also used by Chenghua Lin and Yulan He [23] to detect the sentiment and topics in review.

In the context of large computer science classes, [6] used a lexicon-based approach (VADER) to study how students changed emotions during the semester while taking a large computer science class. They found out that grades and the emotional experience were not correlated thus emphasizing the need to monitor the emotions of students instead of homework or exam performance to ensure a good experience for students in class.

Machine Learning has seen a wide application across various domains including opinion mining and sentiment analysis. Models such as Support Vector Machine, Naive Bayes, and Random Forests have been shown to accurately predict sentiment polarity in Twitter posts [24], hotel reviews [12], movie reviews[10, 11], favorability of politicians based on online activity [25]. Nabeela et. al [16] studied the performance of Naive Bayes, Complement Naive Bayes (CNB), and Support Vector Machine (SVM) models in predicting sentiments of real-time student feedback. They found out that the Support Vector Machine and Complement Naive Bayes had the highest accuracy and the three-class model (negative, neutral, and positive) gave the best performance compared to the two-class (negative and positive) models. In [17], the authors studied the effect of preprocessing and the neutral class on the performance of the Naive Bayes and Support Vector Machine models in predicting sentiment of student feedback. They concluded that model performance was worse with the three-class problem which is in conflict with the findings in [16]. Similarly, Nasim et. al [26] compared a hybrid custom lexicon and machine learning approach to predict sentiments in the student feedback provided at the end of the semester. They found out that the hybrid approach performed better than other text analytics libraries on the internet but did not perform well in capturing negative and neutral sentiments.

Previous work has focused on the use of dedicated raters to label student textual feedback sentiments. In this work, I study the use of labels crowdsourced from other students and students' self-reported Likert scale ratings to train the machine learning models.

Chapter 2

Background

Machine Learning is the process of using a large set of data to learn patterns that can be used to predict a property about individual data points or their source entity. For instance, the set of words and their arrangements in a sentence can be used to predict whether the sentence conveys a negative or positive sentiment. A learning problem may be unsupervised or supervised where each data point in the training is assigned a ground truth label. In a sentiment analysis task, the features should be extracted from the text and the labels may be acquired by using a set of dedicated annotators or employing crowdsourcing.

2.1 Feature Engineering

Machine learning models require a set of features [27] that characterize the properties of a data point in the data to make predictions. In the context of this work, a data point refers to a specific textual homework assignment review provided by a student. Since the reviews are a set of words, there is a need to represent them in form of numeric or categorical features that can be used by the models. This process involves two major stages i.e Preprocessing and Feature extraction.

2.1.1 Preprocessing

This stage cleans and prepares the text for feature extraction - it involves the removal of stop words and unwanted characters such as commas, slashes, apostrophes among others. The

text is also converted to the lower case. Although upper cases are often used to emphasize a sentiment in a word or sentence, other researchers [28] have noted there is no significant difference between the sentiment conveyed by a word in the upper case compared to its lower case. Converting to lower case reduces the feature vector space dimensionality and captures more information when the text is converted to features.

Stop word removal: This involves removing non-sentimental or uninformative words that are only meant to support the construction of sentences in the language. For the English language, words such as "the", "or", "and", "is", "they" among others are considered stop words. These words are removed from the text before doing feature extraction.
Stemming: Stemming involves reducing the individual words to their root form. For example "passed" or "passing" are reduced to "pass".

2.1.2 Term Frequency - Inverse Document Frequency (TF-IDF)

First proposed in 1972 by Spark Jones [29], The Term Frequency - Inverse Document Frequency (TF-IDF) is a method that is widely used in information theory to capture the relevance of the words or phrases contained in a single document in comparison with a set of documents (corpus). The TF-IDF algorithm generates features based on words or phrases in a corpus. The weight assigned to each word or phrase in a given document shows its relevance in the corpus - very common or rare phrases are assigned smaller weights compared to phrases or words that are more informative.
For a specific term t and a document $d \in D$ where D is the total number of documents, the TF-IDF is computed using the formula.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t,d}} idf(t) = log(\frac{N}{|d \in D : t \in d|}) tf - idf(t,d) = tf(t,d) \times idf(t)$$

Where $N = |d|$

When using TF-IDF, features may be based on n-grams [30]. n-grams refer to a continuous set of words in sentences or phrases. Single-word phrases are referred to as "unigrams", two-word phrases are "bi-grams" and three-word phrases are "trigrams". Tri-grams are the

maximum n-grams commonly used since the most common English sentences are at least three words.

2.1.3 Document Embedding

An alternative method of feature generation is Document Embedding. Document Embeddings [31] is based on the Word embedding approach which transforms words in a sentence to a large continuous vector space such that words with similar meanings are closer to each other in the new vector space. Documents embedding represent a whole document in the new vector space. Document embedding is a popular method [32, 33] for generating features from text datasets. The embeddings are learned using a deep neural network trained on large corpora of documents such as Wikipedia or Twitter feeds. Due to the high computational cost of training a document embedding, pre-trained embeddings such as BERT are commonly used instead. Developed by Google, Bidirectional Encoder Representations from Transformers (BERT) [34] is one of the popularly used pre-trained embeddings in Natural Language Processing. The BERT embedding was trained on Wikipedia documents and a dataset containing over 10,000 books from different genres. The large corpora of documents used to train BERT makes it applicable to multiple domains.

2.2 Supervised Machine Learning

A machine learning task may be modeled as a supervised or unsupervised problem. Supervised machine learning methods use ground truth labels for each data point in the training dataset and fit a classifier that maximizes the accuracy of the predicted labels in the training set with the assumption that the model will perform well when tested on unseen data. Unsupervised approaches learn patterns in the training set without any ground truth labels being provided. In this research I used supervised machine learning approaches and the classification problem was set up with three classes i..e $y_i \in \{-1, 0, -1\}$ and $x_i \in \mathbf{R}^{d\times 1}$. I compared the performance of four (4) models namely regularized Linear Model, Support Vector Machine (SVM) Model using a gaussian kernel, Multinomial Naive Bayes, and the

Random Forest. These models have demonstrated the best performance in related sentiment prediction settings [16] as well general machine learning classification problems.

2.2.1 Linear Model

The linear model classifier aims to separate the data using a $d - dimensional$ hyperplane which is a linear combination of the features. Consider the data points $x \in \mathbf{R}^{d\times 1}$ and a corresponding label $y \in \{+1, -1\}$. The linear classifier computes the weights $w \in \mathbf{R}^{d\times 1}$ and computes the prediction using:

$$h(x) = sign(w^T x + b)$$

It learns the optimal weights by minimizing the regularized loss function using the gradient descent (GD) [35] algorithm. To speed up the computation, the stochastic gradient descent (SGD) [36], a variant of gradient descent is used instead when the sample size and the dimensional space are large. In a three-class classification task, the linear model breaks it into two binary classification problems i.e negative (-1) versus positive $(+1)$/neutral (0) and positive $(+1)$ versus neutral (0). The binary classification problems are then solved sequentially.

2.2.2 Support Vector Machine (SVM)

The support vector machine [37] using the Radial Basis Function (RBF) kernel [38] works by transforming the input feature space to an infinite dimension vector. It fits a linear classifier in the new space which maximizes the margin from any point to the linear classifier. It identifies and stores a small sub-set of data points also known as support vectors which are the closest to the classifier in the new space. Predictions are made using only the support vectors. The predictions of the support vector machine can be given by:

$$h(x) = sign(\sum_{i=1,\alpha_i>0}^{n} \alpha_i y_i K(x, x_i) - b)$$

where b is the bias , K is the RBF kernel and α_i are the lagrange multipliers[37] corresponding to the support vectors.

Similar to the Linear Model, the SVM works by considering the classification problem as two binary classification problems which are solved sequentially.

2.2.3 Naive Bayes

The Naive Bayes is a probabilistic model that relies on the assumption of independence between features of the data points given the class label. For any data point $x = \{x_1, x_2, \cdots, x_d\}$, the predicted label $\hat{y}$ is given by

$$\hat{y} = argmax_{c \in \{-1,0,+1\}} P(y = c) \prod_{i=1}^{d} P(x_i | y = c)$$

2.2.4 Random Forest

The random forest is a classifier that aggregates a large ensemble of decision trees [39]. The classification output is the mode of the individual predictions from all the decisions trees in the forest. The decision tree is a non-parametric model and learns simple decision rules from the data and uses them to predict the label of a specific data point. In the random forest, the individual decision trees are constructed using a small randomly selected subset of the features. Although the underlying decision trees may overfit, the random forest is a robust and stable model that does not overfit even when very many trees are used.

2.3 Lexicon Based Methods

Rule-based approaches maintain a lexicon that defines the sentiment orientation associated with individual words or phrases in the English (or any language of interest) dictionary. To predict the sentiment in a given sentence, they aggregate the sentiments of the individual

words or phrases. One commonly used example includes VADER (Valence Aware Dictionary for Sentiment Reasoning) [13] . Other lexicon-based libraries include TexBlob [40, 41] , Flair [42], WordNet [43]. Although these approaches have been demonstrated to perform well in sentiment analysis, they fail to capture the contextual differences as words and phrases may have different meanings and sentiment polarities in different settings. An example in the context of student feedback is how students use the word "problem". A sentence such as "*I like problem one*" would be classified as having a negative sentiment even though it is positive. VADER specifically allows the user to specify new stop words or assign new sentiment values to words in their respective specific context. This manual approach is 1) unreliable since it is hard to exhaust all possible words whose meaning may change and 2) may add bias. A statement such as "*I am having problems with the homework*" is negative but if the word "problem" is redefined as "neutral" in the student feedback context then the statement would be predicted as neutral by VADER.

2.4 Crowdsourcing

Crowdsourcing is a method of gathering information or getting a large number of online users to contribute to a task. With the rise of the internet, this approach has been utilized to cheaply label large datasets such as images [44] as well sentiment analysis datasets [45, 46]. Crowdsourcing experiments are normally set up by asking a set of users to perform a task for a very small remuneration on each unit of the task. Amazon Mechanical Turk (AMT) is a popular platform that has a large set of registered remote workers who are hired to perform tasks such as data labeling [47]. In data labeling tasks, the crowd workers are randomly assigned a single item in the dataset. A data object may receive multiple labels from different workers and these have to be aggregated to get the overall true label.

2.4.1 Majority Vote

The Majority vote label aggregation method assumes that all workers have an equal level of accuracy and thus their labels contribute equally to the final label. The modal label is predicted as the ground truth label[48, 49].

2.4.2 Weighted Majority Vote

Weighted majority vote method acknowledges that different crowd workers will have different accuracies due to experience and thus assigns different weights to the contributions of each worker. Furthermore, some of the workers may be adversarial which can affect the accuracy of the aggregated results [50]. Darwin et. al [51] described a maximum likelihood estimation (MLE) approach to estimating worker accuracies optimized using Expectation-Maximization (EM) algorithm. The EM algorithm starts by assuming accuracies of each labeler, such as an accuracy $p_i = 0.8$ for all labelers (E-step). In the M-step, the algorithm computes the labels of each homework review (E-step) using a weighted majority vote with weights of $w_i = 2p_i - 1$ [52]. The E-step then runs to compute the new accuracies p_i of each worker assuming the computed labels as the ground truth. The algorithm runs until there is no further change in the labels and labeler accuracy.

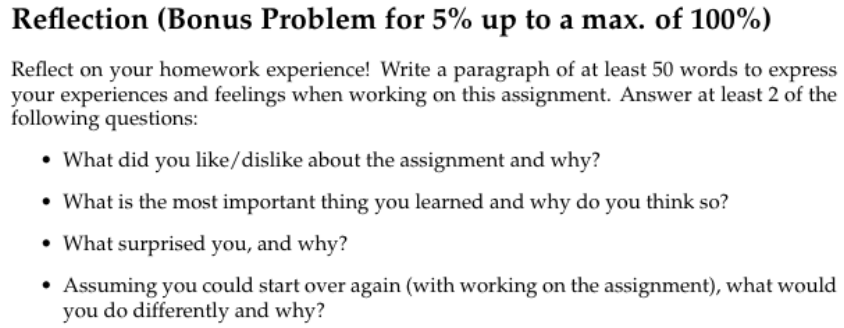
Chapter 3

Methods

3.1 Study Setting and Feedback Data

The dataset used consists of 6023 student homework reflections collected from eight large computer science courses from eight different semesters spanning Spring 2016 to Spring 2020 at a Washington University in St. Louis with institutional approval to study human subjects. Students in these classes were asked to provide feedback about their experience with the homework assignments in exchange for bonus points. When submitting their homework assignment, the students were asked to provide feedback in form of textual reviews of no less than 50 words as well as a star rating (1 to 5) with 1 representing the most negative sentiment about the homework and 5 representing the most positive sentiment about their homework experience. This data was collected for homework assignments during each of the courses. For classes offered before Fall 2018, the students were asked to provide one of the three labels (positive, neutral, and negative) instead of a 5-star rating that was used between Fall 2018 to Spring 2020. Figure 3.1 shows an example prompt that was shown to

the students after submitting their homework assignments.



Reflection (Bonus Problem for 5% up to a max. of 100%)

Reflect on your homework experience! Write a paragraph of at least 50 words to express your experiences and feelings when working on this assignment. Answer at least 2 of the following questions:

- What did you like/dislike about the assignment and why?
- What is the most important thing you learned and why do you think so?
- What surprised you, and why?
- Assuming you could start over again (with working on the assignment), what would you do differently and why?

Figure 3.1: Sample Prompt used by the instructor to ask for feedback

Table 3.1 shows the data collected from the seven offerings of an upper-level cloud computing class and one intro to computer science course. There were fewer students submitting reviews/reflections in Spring 2016 to Fall 2017 classes compared to those who submitted after Spring 2018 when the instructor introduced bonus points on homework grades.

Table 3.1: Sources of the homework reflections

Semester	**Course**	**Number of students**	**Number of Reviews**	**Self-Ratings Available**	**Ground Truth Available**
Spring 2016	Cloud Computing	10	86	86	
Fall 2016	Cloud Computing	13	114	114	
Spring 2017	Cloud Computing	41	348	348	
Fall 2017	Cloud Computing	45	269	269	
Spring 2018	Cloud Computing	97	740	740	
Fall 2018	Cloud Computing	97	722	722	
Fall 2019	Cloud Computing	85	563	563	✓
Spring 2020	Intro to Computer Science	603	3181	3181	

Consistent with the observations by other researchers [53, 54], the student self-ratings were imbalanced with a vast majority of the reviews being positive. Figure 3.2 shows the distributions of the labels across all datasets that were used in this work.

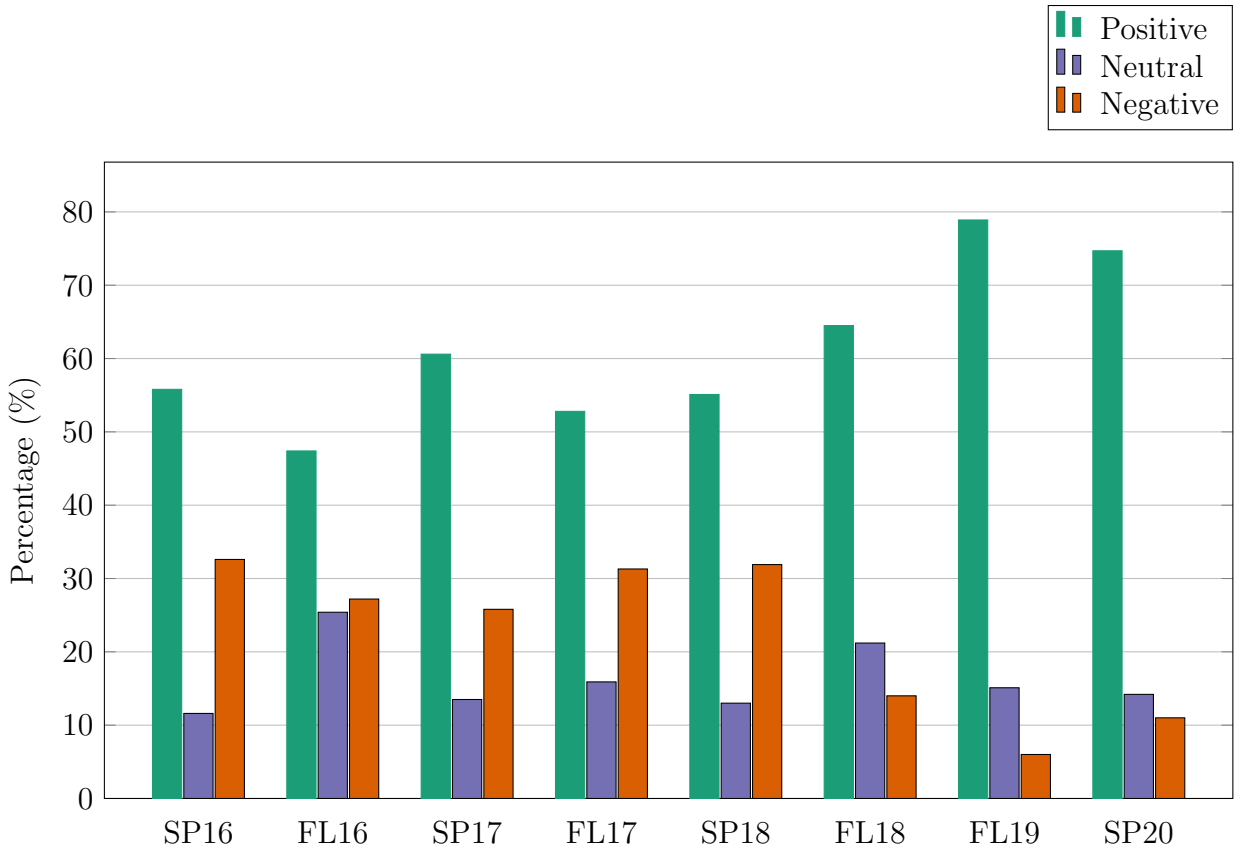


Figure 3.2: Distribution of Labels in the dataset

3.2 Crowdsourced Ground Truth Labels

To establish ground truth labels of the reviews in the dataset, I set up an experiment to collect multiple labels for reviews for the Fall 2019 dataset. I asked the students in an introduction to data science course offered by my research advisor to read and label the de-identified homework reflections. They used a web-based interface shown in Figure 3.3 which was developed to randomly assign and display a homework review and recorded the assigned rating before displaying another review.

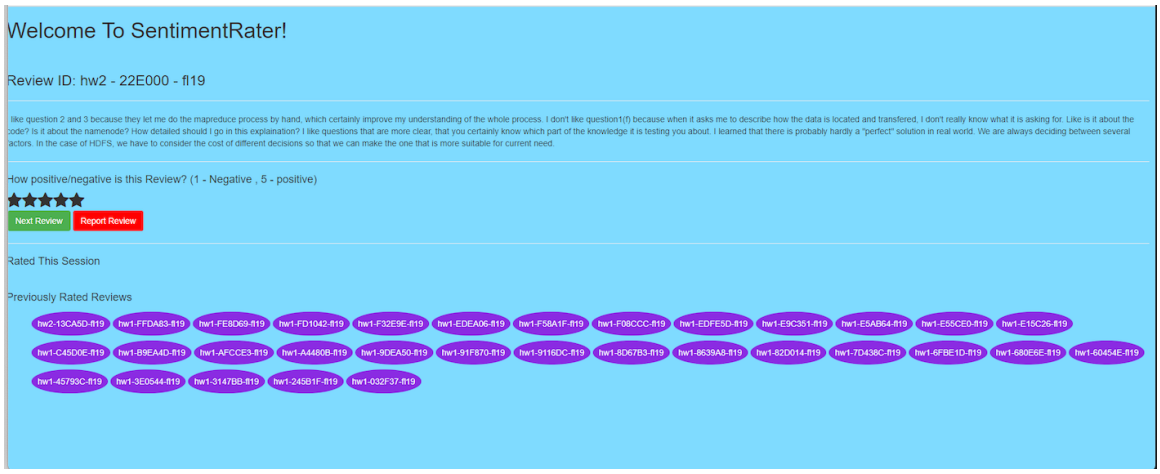


Figure 3.3: Web-based Interface used by the Labelers

For each review, the labelers assigned a five-point Likert scale rating with 1 representing strong negative emotion, 2 representing negative emotion, 3 is neutral, 4 representing positive emotion, and 5 representing strong positive emotions conveyed in the feedback.

To encourage submission quality labels, the students (labelers) were incentivized by getting lab quiz credit. They were further informed that their quiz grade will depend on the number of high-quality labels they produce. For a label to be high-quality, it needed to be within one rating away from the median rating of all other students' ratings. I noticed that the labelers were hesitant to give extreme labels such as 1 and 5 because they are more likely to be more than 1 point away from the median. To address this unexpected challenge, I converted the ratings to a three-point scale [-1,0,1] representing negative, neutral, and positive. Ratings of 1 & 2 were converted to -1, 3 was converted 0 whereas 4&5 were converted to $+1$. I collected a total of 3037 labels from 129 labelers. Each of the 563 reviews received at least three ratings; On average each student labeled 23 reflections and each reflection received five labels.

To determine the true label for each homework review, I experimented with two label aggregation methods that is majority vote and weighted majority.
In a weighted majority vote, the labeler accuracies p_i were estimated using the EM algorithm as described in Section 2.4.2. Below is the algorithm I used to generate the true labels.

1. Initialize all labeler accuracies $p_i = 0.7$
2. Use Expectation Maximization (EM) to estimate labeler accuracies p_i
3. For each review compute the weighted score using weight as $w_i = 2p_i - 1$
4. Predict 0 if the weighted score between -0.1 and $+0.1$, predict -1 if less than -0.1 or predict $+1$ if its greater than $+0.1$

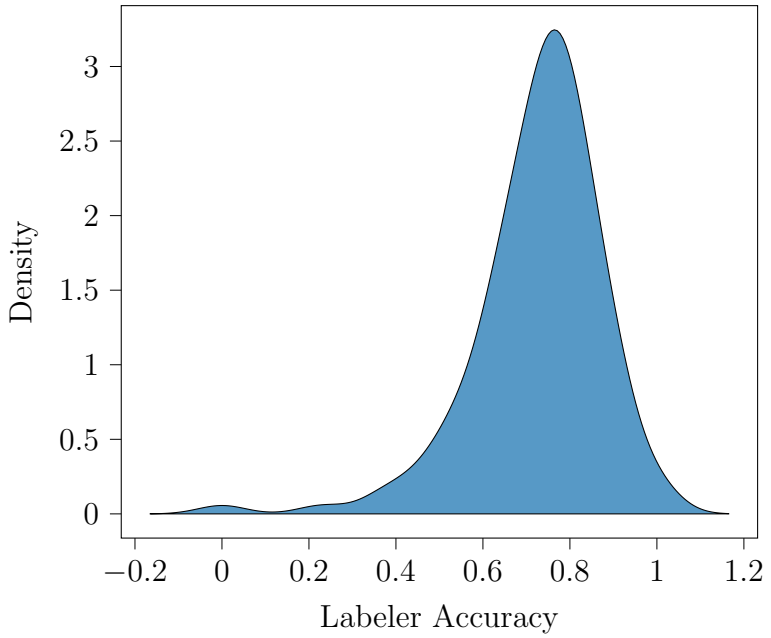


Figure 3.4: Distribution of labeler accuracies estimated EM

Figure 3.4 shows the distribution of labeler accuracies as estimated by the EM algorithm. Out of the 129 labelers, only 3 labelers had a perfect accuracy of 1.0 and 1 labeler had accuracy of 0.0. Majority of labelers (n=100) had an accuracy between 0.58 and 0.88. The modal accuracy was 0.8 whereas the mean accuracy was 0.73. The standard deviation of the distribution was 0.15. Figure 3.5 shows the distribution of the labels obtained using weighted majority vote, majority vote, and the student self-ratings in the Fall 2019 dataset.

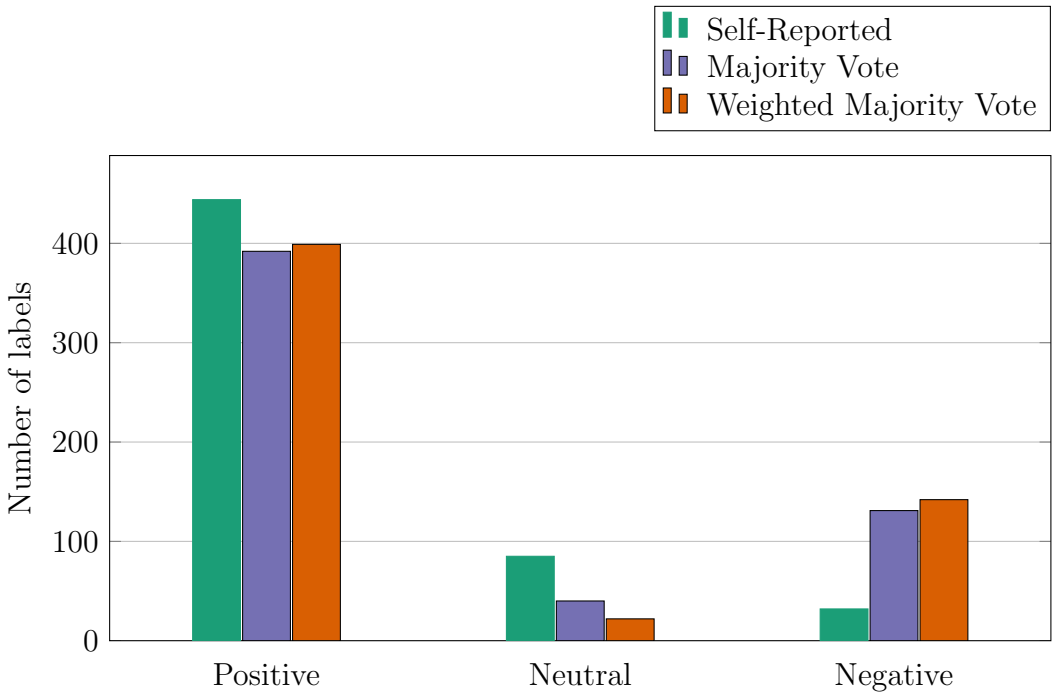


Figure 3.5: Distribution of Labels in the Fall 2019 dataset

3.3 Baseline Setting

For this research, I chose to compare machine learning model performance to VADER as the baseline comparison since it's suited for micro-blog text. When given a piece of text, VADER computes the compound sentiment score as a real number between -1 and $+1$. Since this was a classification problem, I converted the sentiment score into either Negative (-1), Neutral (0), or Positive (+1). VADER's Official Documentation [13] recommends that compound scores between -0.05 to 0.05 should be considered as Neutral (0), compound scores less than -0.05 should be negative (-1) and positive (+1) sentiment corresponds to scores greater than $+0.05$.

In one experiment setting, I did not do any preprocessing of the student homework reviews before computing the VADER score. In another setting, I pre-processed the reviews before using VADER to compute the sentiment prediction. The pre-processing was minimal and aimed at addressing contextual issues that arise due to the use of VADER in the domain of

predicting student feedback. The most obvious contextual challenge was the use of the word "Problem" which is negative in the VADER lexicon yet students and instructors often use statements like "Problem one" to refer to the specific tasks in the homework which should be neutral. I replaced any sequences of "Problem one", "Problem two" etc with "Question one", "Question two" etc since "Question" is neutral in the VADER lexicon. Consider the following example:

Before preprocessing:

> *"I enjoyed working on problem one and two but problem three was confusing. Generally, I did not have any major problems with the homework."*

After preprocessing:

> *"I enjoyed working on question one and two but question three was confusing. Generally, I did not have any major problems with the homework."*

3.4 Model Training

Four supervised machine learning models were chosen for comparison in this study i.e Support Vector Machine, Random Forest, Linear Model, and Naive Bayes. The SciKit Learn package implementations of these models in python were used. The experiments were run on an 8 core CPU using Python 3.7. The training was done using the entire dataset as described in Table 3.1. Only the Fall 2019 dataset was used to evaluate the model performance using cross-validation with 90% of this sub dataset added to the training set and the 10% used for

testing in each fold as shown in Figure 3.6.

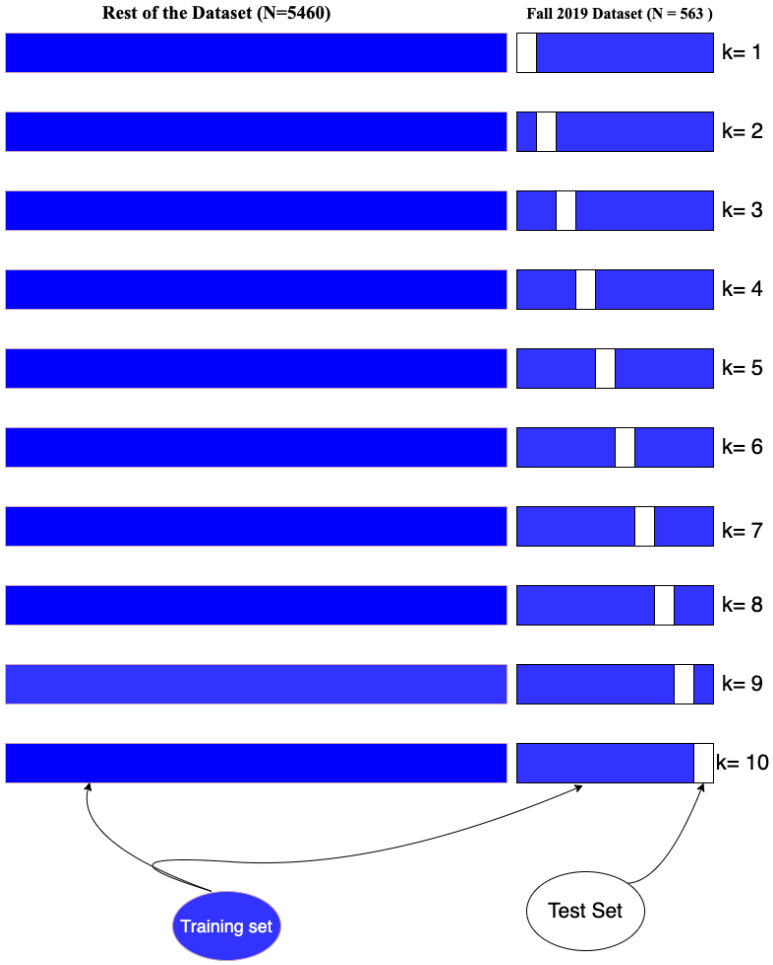


Figure 3.6: Cross-validation Data split

3.4.1 Features

To generate the features, the textual reflections/reviews were preprocessed to remove stop words and to convert them into lower cases. I then generated numeric features for each review using the Term Frequency - Inverse Document Frequency (TF-IDF) and Document Embeddings. Consistent with the observations by [55], the overall performance of the models in preliminary results degraded when stemming was done and thus I decided not to do it for this work. The TF-IDF features were generated using the TfidfVectorizer Library in

SciKit Learn in Python. For TF-IDF, a combination of unigrams, bigrams, and tri-grams was used to generate the features. During the preliminary steps, I experimented with 1) only unigrams, 2) unigrams, and bigrams, and 3) unigrams, bigrams, and tri-grams. A combination of unigrams, bigrams, and trigrams results showed the best performance and thus was used for this work. Since TF-IDF generates very many features, the maximum number of features to use for each model was learned using cross-validation. The training set was used to learn a TF-IDF vectorizer which was also used to transform the textual reflections during testing.

The Document Embedding features were generated using the pre-trained BERT model. In some experiment settings as shown in Table 4.1, the VADER compound scores of the textual reviews were computed and used as an additional feature to the TF-IDF/Document Embedding features.

3.4.2 Hyper Parameter Tuning

The Grid Search Method was used to choose hyper parameters of the models using 10-fold cross validation on the training set optimizing for balanced-accuracy. The SciKit Learn python package implementation of Grid Search was used with hyper parameters shown in Table 3.2 for each model.

Table 3.2: Hyper Parameter search space for each model

Random Forest	
Hyper Parameter	**Values**
max_depth	[16 , 32 , 64 , 128 , 256]
n_estimators	[200 , 300 , 400 , 450 , 500]
class_weight	[balanced , balanced_subsample]
criterion	[entropy , gini]
min_samples_leaf	[7 , 9]
min_samples_split	[7 , 9]
Support Vector Machine	
Hyper Parameter	**Values**
kernel	[linear , poly , rbf , sigmoid]
gamma	[scale , auto]
decision_function_shape	[ovo , ovr]
class_weight	[balanced]

3.4.3 Training Labels

The noisy student self-reported ratings were used as the labels for the homework reflections during the training of the models. The student self-reported ratings were rescaled from 1-5 to three classes (-1,0,+1) representing the negative, neutral and positive sentiment classes. I computed the Mean Absolute difference between 500 randomly samples of labels with the student self-reported ratings (SR), the Majority Vote (MV), and the Weighted Majority Vote (WMV). The results are shown in Table 3.3. The Weighted Majority Vote had the largest Mean Absolute Difference from the random label whereas the self-reported ratings had the smallest.

Although the student self-ratings are noisy, I found them to have a positive correlation to the crowd-sourced labels as shown in Table 3.4. Various experiments were set up to test if these noisy labels would be used to train the models and give a decent performance compared to when only the crowdsourced ground truth labels are used. The advantage of using the student self-labels is that they are readily available since students in the class can provide

Table 3.3: Mean Absolute Difference between Labels

	SR	MV	WMV	Random
SR	-	0.43	0.43	0.949
MV		-	0.05	0.975
WMV			-	0.985
Random				-

Table 3.4: Pearson Correlation coefficients between Labels

	SR	Majority Vote	Weighted MV
SR	-	0.45	0.45
MV		-	0.95
WMV			-

them along with the textual feedback. The ground truth labels obtained from using Majority Vote and Weighted Majority Vote with EM were highly correlated (coefficient = 0.95). I decided to use weighted Majority Vote Labels were the ground truth in this study. Whitehill et. al [49] demonstrated that when you have a relatively large number of labels labeling multiple reviews, the weighted majority vote performs better than the majority votes in estimating true labels. Additionally, the weighted majority vote results had a bigger mean absolute difference from the random label than the majority vote results.

3.5 Performance Evaluation

The evaluation of the model performance was based only on the Fall 2019 dataset since it had ground truth labels obtained from the crowdsource experiment and the weighted majority vote predictions. Reviews from the other datasets were only used in hyperparameter tuning and the training process using the student self-labels as the labels. I used 10-fold stratified cross-validation [56] as shown in Figure 3.6 on the Fall 2019 dataset to evaluate the model performance. Three metrics were chosen for evaluation i.e balanced recall, balanced accuracy, and recall on the negative class. Since the dataset was imbalanced, the balanced accuracy ensures that the models perform well across all sentiment categories.

$$Balanced\ Accuracy = \sum_{c \in \{-1,0,+1\}} P(y=c) \frac{\sum I(\hat{y}=y)}{\sum I(y=c)}$$

If just accuracy is considered then a model would perform well and score 71% accuracy in this dataset by just predicting positive for all reviews since positively labeled. This all-positive model would however only have a balanced accuracy score of only 59%. Maximizing the balanced recall minimizes false negatives in each class. Recall on the negative class computes the proportion of reviews with negative that are classified correctly. The recall for a single class is calculated as

$$Recall = \frac{True\ Positives}{True\ Positives + True\ Negatives}$$

The balanced recall is the weighted average of the recall across all classes. I chose the recall on the negative class as a metric of interest because in this specific research context, I want to minimize cases where students who may be struggling in the course are not given tailored support by the instructor due to their reflections being classified as positive or neutral.

Chapter 4

Results

In this chapter, I describe the performance of VADER and the machine learning models in the various experiment settings that were set up.

4.1 VADER Performance

The lexicon-based VADER algorithm achieved a balanced accuracy score of 73.1% and a balanced recall score of 76.2% whereas the pre-processed version of VADER performed with a balanced accuracy of 73% and a balanced recall score of 76.2%. As shown Figure 4.1 the pre-processed version of VADER was more accurate at capturing positive reflections but worse on negative reflections.

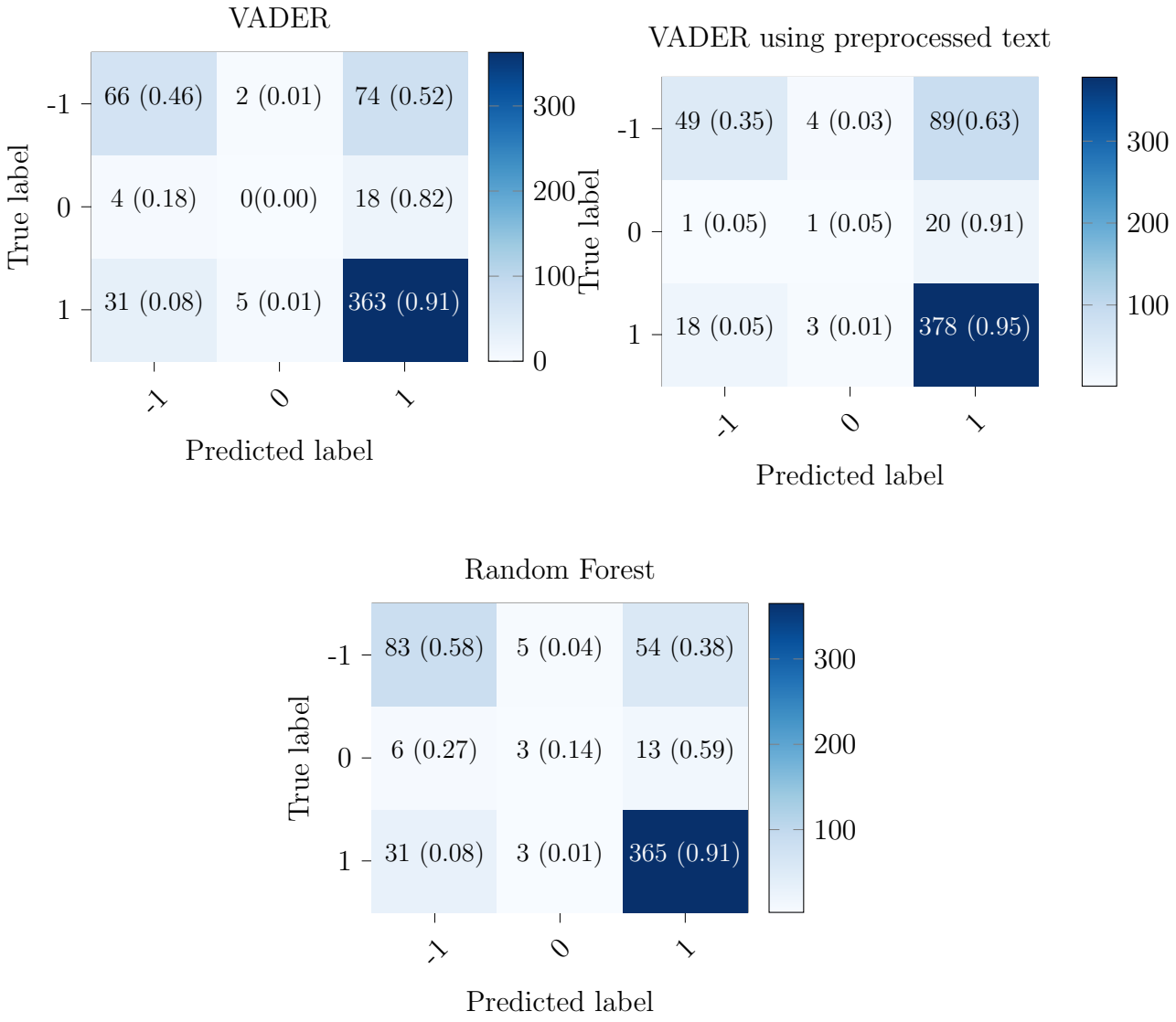


Figure 4.1: Confusion Matrices of VADER and Random Forest

4.2 Machine Learning Model Performance

Preliminary results showed the Naive Bayes and Linear Models performed poorly with balanced accuracy scores around and sometimes less than 50%. Their performance is not

included in these results shown in Table 4.2. Table 4.1 shows the eight (8) different settings that were set up. In experiment settings $1-4$, the TF-IDF features were used as the features where Document Embeddings were used in $5-8$. In experiments, $2, 4, 6, 8$ the VADER scores of the textual reflections were computed and added to the TF-IDF/Document Embedding features. The student self-reported (SR) labels were used as the only training labels in experiments $1, 2, 5, 6$. In experiments $2, 4, 6, 8$, the crowdsourced labels were used as the training labels for reviews from Fall 2019 dataset, and the self-reported labels were used for reviews from the other datasets. In the following sections, I describe the performance of the models in each setting.

Table 4.1: Experiment settings

	Features			Training Labels	
Setting	TF-IDF	Embeddings	VADER score	Self Reported	Crowdsourced (Fall 19 only)
1	✓			✓	
2	✓		✓	✓	
3	✓				✓
4	✓		✓		✓
5		✓		✓	
6		✓	✓	✓	
7		✓			✓
8		✓	✓		✓

Table 4.2: Model Performance in the various settings

	Balanced Recall (%)			Balanced Accuracy (%)			Negative class Recall (%)		
Setting	VD	RF	SVM	VD	RF	SVM	VD	RF	SVM
1	76.2	75.8	70.1	73.1	*73.8*	70.1	46.5	40.0	33.1
2	76.2	*78.3*	73.8	73.1	*76.8*	*73.4*	46.5	46.5	40.1
3	76.2	*76.9*	75.7	73.1	*76.1*	*75.2*	46.5	*56.3*	*51.4*
4	76.2	***79.7***	75.5	73.1	***78.6***	*75.5*	46.5	*58.4*	*52.1*
5	76.2	73.7	53.9	73.1	66.7	60.8	46.5	16.9	*47.1*
6	76.2	75.3	67.6	73.1	69.4	72.2	46.5	23.2	*47.2*
7	76.2	74.8	57.7	73.1	70.8	68.8	46.5	24.6	*50.7*
8	76.2	*77.6*	70.3	73.1	*73.9*	73.1	46.5	36.6	***59.1***

VD = VADER , RF = Random Forest, **Bold** = best score for the metric, *Italic* = better than VADER

In setting 1, the Random Forest model had a slightly higher balanced accuracy than VADER and its performance greatly improved when the VADER score was added as an additional feature in settings 2 and 4. The SVM generally performed worse than the random forest in all experiment settings but got a higher balanced accuracy score than VADER in experiment settings 3 and 4 when crowdsourced labels were used for training. In experiment setting 4, where the VADER score and the crowdsourced labels were used for training, the Random Forest achieved its highest score of 78.6% balanced accuracy and a 58.6% recall on the negative class. Figure 4.1 shows the confusion Matrix of Random Forest in this setting. When using features generated by Document Embeddings (Settings 5-8), the random forest has a higher balanced accuracy and recall scores than the SVM but the SVM model was notably more accurate at predicting the negative sentiments with 59.1% recall.

Effect of training using crowdsourced labels:

To study the effect of using crowdsourced labels on model performance, I varied the percentage of the Fall 2019 dataset with crowdsourced labels in training using TF-IDF and VADER score features. When only student self-reported labels were used (i.e 0% crowdsourced labels), the random forest achieved a balanced accuracy of 76.1%, beating VADER (73%) and SVM (70.1%). As shown in Figure 4.2, the accuracy of both the SVM and Random

Forest models improved as the percentage crowdsourced labels used increased with Random Forest and SVM achieving 76.1% and 75.5% balanced accuracies respectively when only crowdsourced labels were used.

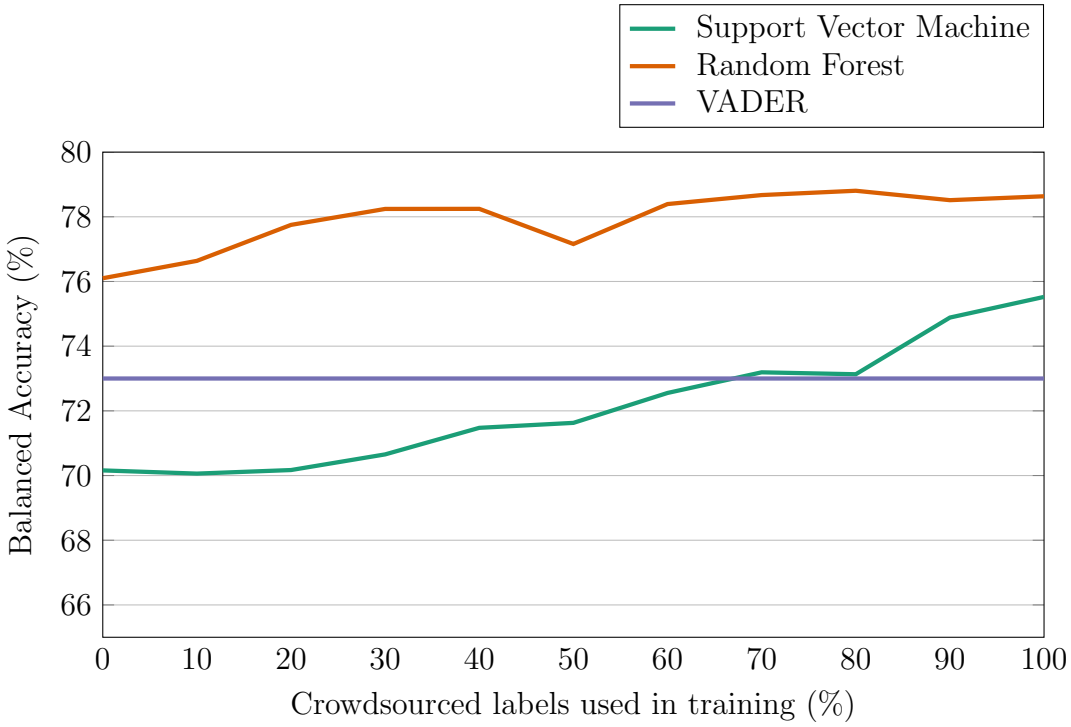


Figure 4.2: Performance with varying percentage of crowdsourced labels

Chapter 5

Discussion and Conclusions

5.1 Discussion

5.1.1 Sentiment Prediction

VADER performed better at identifying homework reviews with negative sentiment when the reviews were not preprocessed than when they were preprocessed cf. Figure 4.1. Preproccessing improved VADER's accuracy at predicting reviews with positive sentiments. This can be explained by the fact that in the preprocessed reviews, the word "problem" was replaced with "question" whenever it referred to a task in the homework which reduced the number of reviews being predicted as negative by VADER. The Random Forest outperformed VADER in most settings achieving it's best performance when TF-IDF features and VADER scores were used as features. The Support Vector Machine (SVM) also outperformed VADER in this setting but performed worse than the Random Forest. When trained solely only on the noisy student self-ratings, the machine learning models did not show any advantages over using VADER. Adding the VADER score as a feature, greatly improved the performance of the models with the Random Forest outperforming VADER alone as well as the Support Vector Machine. The Random Forest performed well since it's more robust to noise than the Support Vector Machine. This robustness can be attributed to the large ensemble of decision trees in the forest and thus can handle noise [39] in the self-reported labels. Using quality ground truth labels reduces the noise in the labels and thus the models performance improved when the crowdsourced labels were used in settings $3, 4, 7 and 8$.. Generally, there

were multiple settings in which machine learning models beat the lexicon-based method VADER thus answering the first research question.

Although the data used to train the models was collected from *different* large-scale computing classes, the results from this study demonstrate that good performance can be achieved since the homework reviews come from a similar context. I learned that even when using the student self-reported labels only, increasing the number of training reviews made the models especially the Random Forest perform better. Varying the percentage of reviews with the crowdsourced labels showed that the model's performance improved with more crowdsourced labels used in training. In this study, I only collected ground truth labels for just 563 reviews in the Fall 2019 dataset but with a larger training set with crowdsourced labels the performance of the model would likely improve further. The models did not perform well with the features generated by document embeddings and were generally worse than when trained on TF-IDF features. This result is surprising; I believe it's because the BERT embedding used in this work was not trained on the student feedback dataset and thus lacks the context information. Overall, I found out that the machine learning models achieved the best performance when trained in the hybrid setting (TF-IDF and VADER scores as features) and crowdsourced labels which answers the second research question.

5.1.2 Crowdsourcing Experiment as a Learning Opportunity

In addition to providing an efficient way of collecting ground truth labels, the crowdsourcing experiment can be used as a learning activity for students. I asked a different student population that had not been asked to provide reflections themselves to participate in our crowdsourcing experiment. The activity was part of the last lab in an intro-level data science course taught by my research advisor. She chose the end of the semester for this activity so that students could see it as reminder at the end of the course that data isn't something that comes in a spreadsheet, but needs to be collected. At this point, they already knew sentiment analysis and supervised classification, which got introduced at an earlier time in the semester.

A real-world crowdsourcing experiment is a great opportunity to illustrate to students that data collection is challenging as well as to introduce crowdsourcing to them in a very hands-on way. Here are some student comments on the activity that show the effectiveness of this learning activity:

"*I learned what crowdsourcing is and how it works. I had heard of it before but never knew what it actually was, and now I do!*"

"*I learned about crowdsourcing and the difficulty of getting labeled data to train models.*"

"*I got a chance to try crowdsourcing data, so I learned about one of the ways data scientists can collect data in a scalable way.*"

In addition to the crowdsourcing experiment, she introduced the specific task in this work, the prediction of student emotions from textual feedback, so that they could learn about a real-world application of sentiment analysis they could immediately relate to themselves. One student reported: "*I learned about crowdsourcing and sentiment analysis and how it is implemented in real-world experiments*"

Further, this activity can show students that collecting and labeling data is not trivial and care needs to be taken when annotating data. Some student's reported on this realization:

"*I learned about the value of crowdsourcing and putting in the time to give honest ratings.*"

"*I [...] learned that in terms of crowdsourced labeling, the incentive needs to ensure high-quality labeling, rather than completion and that there is a reasonably easy way of judging quality (look at the median score for every score and only take a certain standard deviation).*"

All students' comments cited above are (parts of) responses to the lab quiz question: "*What is the one thing you learned from today's lab? Write 1-2 full sentences.*" This question was part of every lab quiz in this course.

5.2 Conclusions and Future Work

Through this work, I have demonstrated that machine learning approaches such as Random Forest and Support Vector Machine can be used to identify sentiments in student unit of study reflections better than a lexicon-based approach alone. Overall, our hybrid approach using TF-IDF and VADER features achieved an accuracy of almost 80% on the sentiment prediction task, this is quite impressive given that sentiments are subjective even to humans. Although student self-ratings are noisy and often biased, they can be useful in training the machine learning models. For higher performance, quality ground truth labels collected from a crowdsourcing experiment can be used to train the model. For an instructor who wishes to use this method to identify students having a negative emotional experience, they can easily set up the experiment by collecting data from any class they or one of their colleagues teach and crowdsource the ground truth labels from another student population. This experiment can serve as a learning opportunity for students in data science or machine learning courses. Finally, to use the predictor as a *live* tool for the instructor to trigger interventions, it is sufficient to use data from previous offerings of the specific course or of similar courses to train the ML model.

In future work, it would be important to improve recall on the negative class, since I believe that one of the most beneficial use-cases for this work is to identify students, teams, or course materials at risk. One way to achieve this could be to improve the crowdsourcing experiment by investigating how many labels per review are necessary for optimal machine learning model performance. This work can further be extended by crowdsourcing ground truth labels on reviews from more courses to investigate if the machine learning models would be able to achieve a balanced accuracy greater than 80%. The envisioned application of this work and is the use of this approach to predict student emotions from written feedback as a means to help course instructors trigger interventions or improve course materials in a classroom study. Studying the effectiveness of this intervention method in a classroom study will be an important future work.

References

[1] Prashant Loyalka, Ou Lydia Liu, Guirong Li, Igor Chirikov, Elena Kardanova, Lin Gu, Guangming Ling, Ningning Yu, Fei Guo, Liping Ma, et al. Computer science skills across china, india, russia, and the united states. *Proceedings of the National Academy of Sciences*, 116(14):6732–6736, 2019.

[2] Kathleen J Lehman, Julia Rose Karpicz, Veronika Rozhenkova, Jamelia Harris, and Tomoko M Nakajima. Growing enrollments require us to do more: Perspectives on broadening participation during an undergraduate computing enrollment boom. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 809–815, 2021.

[3] Engineering National Academies of Sciences, Medicine, et al. *Assessing and responding to the growth of computer science undergraduate enrollments*. National Academies Press, 2018.

[4] Kai Presler-Marshall, Sarah Heckman, and Kathryn T Stolee. Identifying struggling teams in software engineering courses through weekly surveys. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, pages 126–132, 2022.

[5] Niki Gitinabard, Sarah Heckman, Tiffany Barnes, and Collin Lynch. Designing a dashboard for student teamwork analysis. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, pages 446–452, 2022.

[6] Marion Neumann and Robin Linzmayer. Capturing student feedback and emotions in large computing courses: A sentiment analysis approach. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021.

[7] Robert L Holzbach. Rater bias in performance ratings: superior, self-, and peer ratings. *Journal of applied psychology*, 63(5):579, 1978.

[8] Daniel Leising, Kenneth D Locke, Elena Kurzius, and Johannes Zimmermann. Quantifying the association of self-enhancement bias with self-ratings of personality and life satisfaction. *Assessment*, 23(5):588–602, 2016.

[9] KEVIN R MURPHY. Modesty bias in self-ratings of performance: A test of the cultural relativity hypothesis. *Personnel Psychology*, 46(2):357–363, 1993.

[10] Kuat Yessenov and Saša Misailovic. Sentiment analysis of movie review comments. *Methodology*, 17:1–7, 2009.

[11] Reza Maulana, Panny Agustia Rahayuningsih, Windi Irmayani, Dedi Saputra, and Wanty Eka Jayanti. Improved accuracy of sentiment analysis movie review using support vector machine based information gain. In *Journal of Physics: Conference Series*, volume 1641, page 012060. IOP Publishing, 2020.

[12] Victor Chang, Lian Liu, Qianwen Xu, Taiyu Li, and Ching-Hsien Hsu. An improved model for sentiment analysis on luxury hotel review. *Expert Systems*, page e12580, 2020.

[13] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

[14] Christopher SG Khoo and Sathik Basha Johnkhan. Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons. *Journal of Information Science*, 44(4):491–511, 2018.

[15] Akshi Kumar and Geetanjali Garg. Systematic literature review on context-based sentiment analysis in social multimedia. *Multimedia tools and Applications*, 79(21):15349–15380, 2020.

[16] Nabeela Altrabsheh, Mihaela Cocea, and Sanaz Fallahkhair. Learning sentiment from students' feedback for real-time interventions in classrooms. In *International conference on adaptive and intelligent systems*, pages 40–49. Springer, 2014.

[17] Nabeela Altrabsheh, Mihaela Cocea, and Sanaz Fallahkhair. Sentiment analysis: towards a tool for analysing real-time students feedback. In *2014 IEEE 26th international conference on tools with artificial intelligence*, pages 419–423. IEEE, 2014.

[18] Venugopal Dhanalakshmi, Dhivya Bino, and Abinaya M Saravanan. Opinion mining from student feedback data using supervised learning algorithms. In *2016 3rd MEC international conference on big data and smart city (ICBDSC)*, pages 1–5. IEEE, 2016.

[19] Hannah J.E. Bijlsma, Adrie J. Visscher, Marjoleine J. Dobbelaer, and Bernard P. Veldkamp. Does smartphone-assisted student feedback affect teachers' teaching quality? *Technology, Pedagogy and Education*, 28(2):217–236, 2019.

[20] Niki Gitinabard, Yiqiao Xu, Sarah Heckman, Tiffany Barnes, and Collin F. Lynch. How widely can prediction models be generalized? performance prediction in blended courses. *IEEE Transactions on Learning Technologies*, 12(2):184–197, 2019.

[21] Maija Hujala, Antti Knutas, Timo Hynninen, and Heli Arminen. Improving the quality of teaching by utilising written student feedback: A streamlined process. *Computers & Education*, 157:103965, 2020.

[22] Swapna Gottipati, Venky Shankararaman, and Jeff Lin. Latent dirichlet allocation for textual student feedback analysis. 2018.

[23] Chenghua Lin, Yulan He, Carlos Pedrinaci, and John Domingue. Feature lda: a supervised topic model for automatic detection of web api documentations from the web. In *International Semantic Web Conference*, pages 328–343. Springer, 2012.

[24] Sunir Gohil, Sabine Vuik, Ara Darzi, et al. Sentiment analysis of health care tweets: review of the methods used. *JMIR public health and surveillance*, 4(2):e5789, 2018.

[25] Vikash Nandi and Suyash Agrawal. Political sentiment analysis using hybrid approach. *International Research Journal of Engineering and Technology*, 3(5):1621–1627, 2016.

[26] Zarmeen Nasim, Quratulain Rajput, and Sajjad Haider. Sentiment analysis of student feedback using machine learning and lexicon based approaches. In *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*, pages 1–6, 2017.

[27] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists.* " O'Reilly Media, Inc.", 2018.

[28] Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information processing & management*, 50(1):104–112, 2014.

[29] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[30] Alexander M Robertson and Peter Willett. Applications of n-grams in textual information systems. *Journal of Documentation*, 1998.

[31] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[32] Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. Linked document embedding for classification. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 115–124, 2016.

[33] Zhaocheng Zhu and Junfeng Hu. Context aware document embedding. *arXiv preprint arXiv:1707.01521*, 2017.

[34] Shivaji Alaparthi and Manit Mishra. Bert: A sentiment analysis odyssey. *Journal of Marketing Analytics*, 9(2):118–126, 2021.

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

[36] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[37] Hui Xue, Qiang Yang, and Songcan Chen. Svm: Support vector machines. *The top ten algorithms in data mining*, 6(3):37–60, 2009.

[38] Vasileios Apostolidis-Afentoulis and Konstantina-Ina Lioufi. Svm classification with linear and rbf kernels. *July): 0-7. http://www. academia. edu/13811676/SVM Classification with Linear and RBF kernels.[21]*, 2015.

[39] Andrew Sage. Random forest robustness, variable importance, and tree aggregation. 2018.

[40] Steven Loria et al. textblob documentation. *Release 0.15*, 2:269, 2018.

[41] J Praveen Gujjar and Prasanna Kumar HR. Sentiment analysis: Textblob for decision making. *Int. J. Sci. Res. Eng. Trends*, (7):1097–1099, 2021.

[42] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

[43] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[44] Anne Grote, Nadine S Schaadt, Germain Forestier, Cédric Wemmert, and Friedrich Feuerhake. Crowdsourcing of histological image labeling and object delineation by medical students. *IEEE transactions on medical imaging*, 38(5):1284–1294, 2018.

[45] Mahsa Heidari and Pirooz Shamsinejad. Producing an instagram dataset for persian language sentiment analysis using crowdsourcing method. In *2020 6th International Conference on Web Research (ICWR)*, pages 284–287. IEEE, 2020.

[46] Elena Filatova. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *Lrec*, pages 392–398. Citeseer, 2012.

[47] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[48] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(43):1297–1322, 2010.

[49] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems*, 22, 2009.

[50] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Identifying unreliable and adversarial workers in crowdsourced labeling tasks. *The Journal of Machine Learning Research*, 18(1):3233–3299, 2017.

[51] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

[52] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *International Conference on Machine Learning*, pages 534–542. PMLR, 2013.

[53] Kushankur Ghosh, Arghasree Banerjee, Sankhadeep Chatterjee, and Soumya Sen. Imbalanced twitter sentiment analysis using minority oversampling. In *2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)*, pages 1–5, 2019.

[54] Dongmei Zhang, Jun Ma, Jing Yi, Xiaofei Niu, and Xiaojing Xu. An ensemble method for unbalanced sentiment classification. In *2015 11th international conference on natural computation (ICNC)*, pages 440–445. IEEE, 2015.

[55] Aditya Wiha Pradana and Mardhiya Hayaty. The effect of stemming and removal of stopwords on the accuracy of sentiment analysis on indonesian-language texts. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pages 375–380, 2019.

[56] Daniel Berrar. Cross-validation., 2019.

[57] Vipul Pandey and C Iyer. Sentiment analysis of microblogs. *CS 229: Machine learning final projects*, 2009.

[58] Andrew Christian Flores, Rogelyn I. Icoy, Christine F. Peña, and Ken D. Gorro. An evaluation of svm and naive bayes with smote on sentiment analysis data set. In *2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*, pages 1–4, 2018.

Student Feedback Emotion Prediction, Kasumba, M.S. 2022