Continuous Faculty Feedback with Sentiment Analysis

Abstract:
Most of the time in educational institutions, faculty feedback is taken either once a year or once in a semester. This feedback collectively shows how the faculty has taught that particular subject over the year, which may be of useful information to the management. The idea is rather than giving feedback to the faculty once a year/semester. A continuous feedback system will help the faculty to assess their performance through which they can improve or increase their level for some specific units or chapters of the subject. The student will give the desired feedback in the form of text to the application, and the application will then use a machine learning model to adjust the score of that desired faculty teaching that subject. The textual feedback, typically collected towards the end of a semester, provides useful insights into the overall teaching quality and suggests valuable ways for improving teaching methodology over the traditional method of taking feedback.

Keywords:
Continuous Feedback System, Machine Learning, Sentiment Analysis

Introduction:

Evaluation of a class and the instructor by students towards the end of each semester has now become a norm in higher education institutions for this in any schools/colleges students are often asked to give feedback to the faculty who have taught that subject either at the end of the year or at the end of the semester. This feedback is either on the scale of 0 to 5 or based on stars. This information is necessary for the management to decide about the usefulness of the faculty. This feedback also serves as one of the necessary purposes for the institution management to decide whether they must continue the same faculty for the respective subject or not.

Another big reason for this project proposal is that the student may feel hesitant if he/she is in a pack of 50-60 students to stand up and raise a question to the faculty during the class hours. Some students may feel that the faculty is teaching some of the chapters in a rushed manner and he/she may not be able to understand the course. In such cases, the student may feel hesitant to tell the faculty to slow down or teach in a much more explained manner.

So, hence in this project proposed a concept that students can continuously provide their feedback based on the teaching method of the faculty during the entire course given any day and any time. Based on this feedback the faculty are provided with a score where they may be able to view their scores changing based on the feedback given by the students. This provides more flexibility for the student as they may anonymously affect the score of the faculty during a chapter/unit. This helps the faculty to realize that some of their class students are facing problems in the ongoing chapter/ unit, who will then slow down and teach that particular subject in a much more detailed manner.

Since machine learning is such a hot topic nowadays, every application in the market leverages the benefits of machine learning. It is the contextual mining of text which identifies and extracts subjective information in the source material and helps a business to understand the social sentiment of their brand, product or service while monitoring online conversations. However, analysis of social media streams is usually restricted to just basic sentiment analysis and count-based metrics. Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative or neutral.

This project aims at identifying the sentiment polarity expressed in textual feedback by a student. The paper employs a hybrid approach to build a predictive model for sentiment analysis.

Literature Review:

The authors of [1], have done research on the classification of emotions of tweets' data gathered from Twitter..they have used ensemble machine learning techniques combining support vector machines with decision trees. Based on the tests conducted, they have acquired a better score in terms of f-measure and accuracy. The main aim of this research was to increase the reliability of the model.

In [2], they utilize the naive Bayes and fuzzy Classifier to classify Tweets into positive, negative or neutral behavior of a particular person. They have presented an experimental evaluation of our dataset and classification results which proved that the combined proposed method is more efficient in terms of Accuracy, Precision and Recall.

In [3], they have presented a model called SuFo(Student Online Feedback System). The model evaluation is based on students' responses in SUFO that measured the lecturers' ability based on the lecturer's professionalism and teaching methods. The results indicated from the student feedback on teaching quality of experience and inexperience lecturers is inconclusive since both categories of lecturers obtained low and also high ratings from students. This model has been used in a public university in Malaysia.

The authors of [4], argue that Semantic Web (SW) technology is a promising approach for data selection and retrieval. The main focus in this paper is to extract knowledge from the feedback given by the students and this can be done by firing Sparql Query in Ontology. This knowledge can then be represented in a meaningful form.

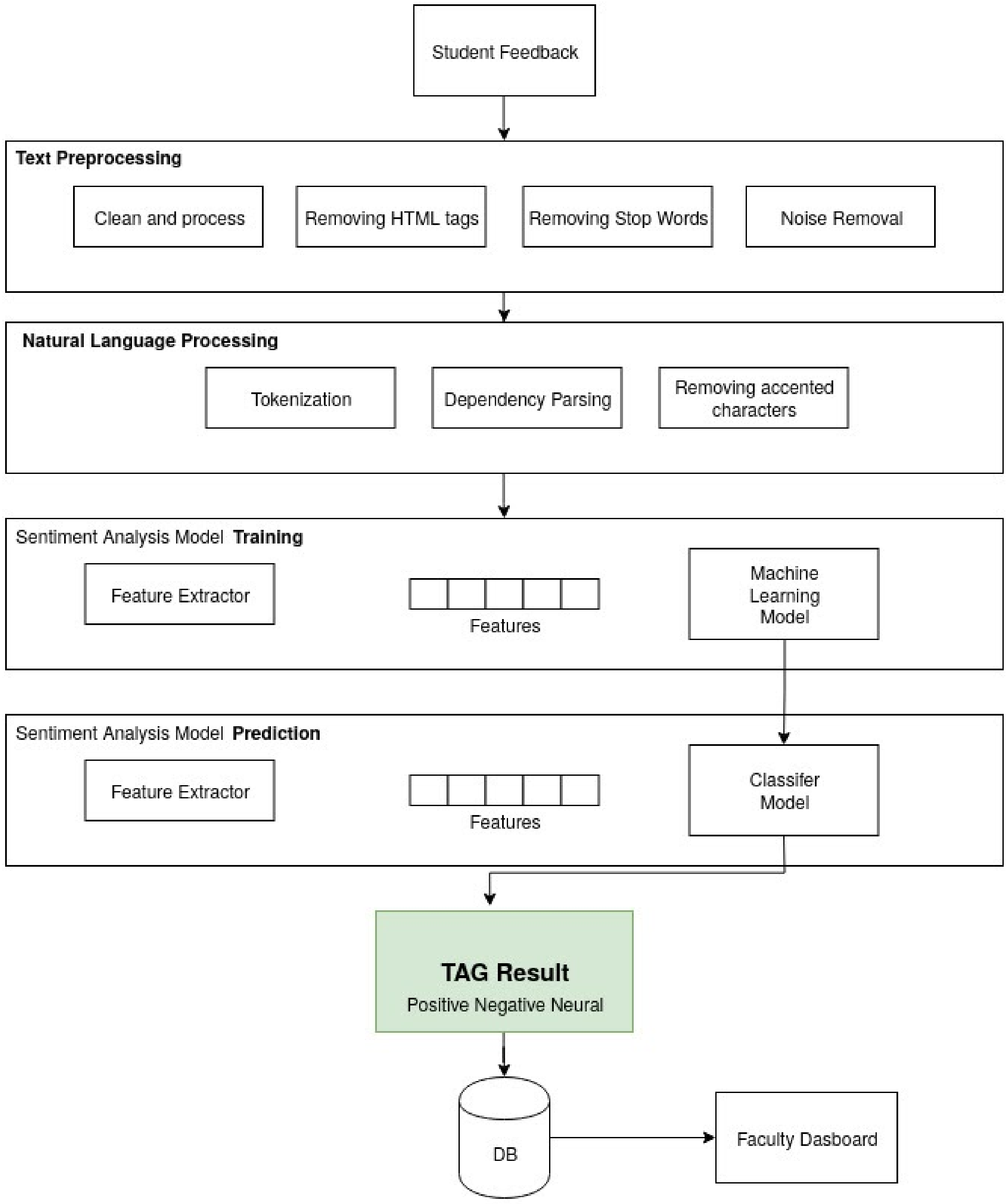
Fernandez et al. [5] presented a method of computing semantic orientation of unstructured text-based on dependency parsing technique. The proposed method leveraged the use of sentiment lexicons which were created using a semi-automatic polarity expansion algorithm. Supervised machine learning approaches of sentiment analysis involve training classifiers using linguistic features that are extracted from the text.

Altrabsheh et al. [6] presented a supervised learning approach to predict sentiment from students' feedback. The reported models were trained using n-gram features extracted from the feedback text. Naive Bayes, Maximum Entropy and Support Vector Machine (SVM) algorithms were used to train models.

In the educational domain, most of the researchers have collected data from social media networks such as Facebook and Twitter. A common preprocessing technique is identifying emoticons, which is found in Troussas et al. [9]. Another preprocessing technique used widely with educational domains is the spelling check; this is found in Ortigosa et al. [8] and Martin et al. [9]. Most of the researchers have used the general preprocessing methods listed above as well. It has been argued that the neutral class is needed in real-life applications [6], [9], including education [4], [10]. Students may have positive, negative or neutral opinions. Discarding the neutral class and focusing only on positive and negative opinions does not show a complete picture of the class opinion and may distort the true proportions of the positive and negative opinions when the neutral proportion is not known. Many of the researchers have included the neutral class when analyzing data from the educational domain [4], [9]. Consequently, the role of a neutral class deserves further investigation.

[14]Go et al. (2009) use distant learning to acquire sentiment data. They use tweets ending in positive emoticons like “:)” “:-)” as positive and negative emoticons like “:(” “:-(” as negative. They build models using Naive Bayes, MaxEnt and Support Vector Machines (SVM), and they report SVM outperforms other classifiers. In terms of feature space, they try a Unigram, Bigram model in conjunction with parts-of-speech (POS) features. They note that the unigram model outperforms all other models. Specifically, bigrams and POS features do not help.
[15] Pak and Paroubek (2010) collect data following a similar distant learning paradigm. They perform a different classification task though: subjective versus objective. For subjective data they collect the tweets ending with emoticons in the same manner as Go et al. (2009). For objective data they crawl twitter accounts of popular newspapers like “New York Times”, “Washington Posts” etc. They report that POS and bigrams both help (contrary to results presented by Go et al. (2009)). Both these approaches, however, are primarily based on n-gram models. Moreover, the data they use for training and testing is collected by search queries and is therefore biased. In contrast, we present features that achieve a significant gain over a unigram baseline. In addition we explore a different method of data representation and report significant improvement over the unigram models. Another contribution of this paper is that we report results on manually annotated data that does not suffer from any known biases. Our data is a random sample of streaming tweets unlike data collected by using specific queries. The size of our hand-labeled data allows us to perform cross-validation experiments and check for the variance in the performance of the classifier across folds.

Proposed Architecture:



The architecture for analyzing the feedback/comments provided by the student for faculty consist of 3 basic blocks:

- Text Pre-Processing
- Natural language Processing
- Sentiment Analysis Model

Text processing:

Text preprocessing is as a matter, of course, the most vital and first step for natural language processing (NLP) jobs. It transforms text into a more digestible form so that machine learning algorithms can perform better. Text is just a sequence of words, or more precisely, a sequence of characters. But when we usually deal with language modeling, or natural language processing, we are more concerned about the words as a whole, instead of just worrying about character-level depth of our text data. One reason behind that is, that in the language models, individual characters don't have a lot of "context". Characters like 'd', 'r', 'a', 'e' doesn't hold any context individually, but when rearranged in the form of a word, they might generate the word "read", which might explain some activity you're probably doing right now.

Text Preprocessing involves some basic and advanced techniques:

- Clean and process
  Cleaning involves, removing HTML tags, punctuation symbols, removing whitespaces, Convert accented characters to ASCII characters, Convert accented characters to ASCII characters
- Tokenization
  Tokenization is a step that splits longer strings of text into smaller pieces or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc.
- Normalizing
  Normalization generally refers to a series of related tasks meant to put all text on a level playing field: converting all text to the same case (upper or lower), removing punctuation, converting numbers to their word equivalents, and so on. Normalization puts all words on equal footing and allows processing to proceed uniformly. Normalizing includes the following tasks:
    - Stemming
    - Lemmatization
    - Word Counts
    - Spell and Grammar Correction
- Removing Stop Words
- Noise Removal
  Noise removal continues the substitution tasks of the framework. While the first 2 major steps of our framework (tokenization and normalization) were generally applicable as-is

to nearly any text chunk or project (barring the decision of which exact implementation was to be employed or skipping certain optional steps, such as sparse term removal, which simply does not apply to every project), noise removal is a much more task-specific section of the framework.

Clean and Process:

The raw text is pretty messy for these reviews so before we can do any analytics we need to clean things up.

Removing HTML tags and special characters:

```
import re
REPLACE_NO_SPACE = re.compile("[.;:!\'?,\"()\[\]]")
REPLACE_WITH_SPACE = re.compile("(<br\s*/><br\s*/>)|(\-)|(\/)")

def preprocess_reviews(reviews):
      reviews = [REPLACE_NO_SPACE.sub("", line.lower()) for line in reviews]
      reviews = [REPLACE_WITH_SPACE.sub(" ", line) for line in reviews]
      return reviews
reviews_train_clean = preprocess_reviews(reviews_train)
reviews_test_clean = preprocess_reviews(reviews_test)
```

Normal Text	Text After basic cleaning
"This isn't the comedic Robin Williams, nor is it the quirky/insane Robin Williams of recent thriller fame. This is a hybrid of the classic drama without over-dramatization, mixed with Robin's new love of the thriller. But this isn't a thriller, per se. This is more a mystery/suspense vehicle through which Williams attempts to locate a sick boy and his keeper.<br /><br />Also starring Sandra Oh and Rory Culkin, this Suspense Drama plays pretty much like a news report until William's character gets close to achieving his goal.<br /><br />I must say that I was highly entertained, though this movie fails to teach, guide, inspect, or amuse. It felt more like I was watching a guy (Williams), as he was	"this isnt the comedic robin williams nor is it the quirky insane robin williams of recent thriller fame this is a hybrid of the classic drama without over dramatization mixed with robins new love of the thriller but this isnt a thriller per se this is more a mystery suspense vehicle through which williams attempts to locate a sick boy and his keeper also starring sandra oh and rory culkin this suspense drama plays pretty much like a news report until williams character gets close to achieving his goal i must say that i was highly entertained though this movie fails to teach guide inspect or amuse it felt more like i was watching a guy williams as he was actually performing the actions from a third person perspective in

actually performing the actions, from a third-person perspective. In other words, it felt real, and I was able to subscribe to the premise of the story.<br /><br />All in all, it's worth a watch, though it's definitely not Friday/Saturday night fare.<br /><br />It rates a 7.7/10 from...<br /><br />the Fiend :."	other words it felt real and i was able to subscribe to the premise of the story all in all its worth a watch though its definitely not friday saturday night fare it rates a from the fiend"

Removing Stop Words:

```
from nltk.corpus import stopwords

english_stop_words = stopwords.words('english')
def remove_stop_words(corpus):
    removed_stop_words = []
    for review in corpus:
        removed_stop_words.append(
            ' '.join([word for word in review.split()
                      if word not in english_stop_words])
        )
    return removed_stop_words

no_stop_words = remove_stop_words(reviews_train_clean)
```

Before Stopwords	After removing Stopwords
"bromwell high is a cartoon comedy it ran at the same time as some other programs about school life such as teachers my years in the teaching profession lead me to believe that bromwell high’s satire is much closer to reality than is teachers the scramble to survive financially the insightful students who can see right through their pathetic teachers’ pomp the pettiness of the whole situation all remind me of the schools i knew and their students when i saw the episode in which a student repeatedly tried to burn down the school i immediately recalled at high a classic line inspector i’m here to sack one of your teachers student welcome to bromwell high i	"bromwell high cartoon comedy ran time programs school life teachers years teaching profession lead believe bromwell high's satire much closer reality teachers scramble survive financially insightful students see right pathetic teachers' pomp pettiness whole situation remind schools knew students saw episode student repeatedly tried burn school immediately recalled high classic line inspector i'm sack one teachers student welcome bromwell high expect many adults age think bromwell high far fetched pity"

expect that many adults of my age think that bromwell high is far fetched what a pity that it isn’t"	

Normalization

Text normalization is the process of transforming text into a single canonical form that it might not have had before. Normalizing text before storing or processing it allows for separation of concerns since the input is guaranteed to be consistent before operations are performed on it. Text normalization requires being aware of what type of text is to be normalized and how it is to be processed afterward; there is no all-purpose normalization procedure.
Two methods that exist for this are Stemming and Lemmatization.

Considering the example text for normalization:

"this is not the typical mel brooks film it was much less slapstick than most of his movies and actually had a plot that was followable leslie ann warren made the movie she is such a fantastic underrated actress there were some moments that could have been fleshed out a bit more and some scenes that could probably have been cut to make the room to do so but all in all this is worth the price to rent and see it the acting was good overall brooks himself did a good job without his characteristic speaking to directly to the audience again warren was the best actor in the movie but fume and sailor both played their parts well"

Stemming

Stemming is considered to be the more crude/brute-force approach to normalization (although this doesn’t necessarily mean that it will perform worse). There are several algorithms, but in general, they all use basic rules to chop off the ends of words.

```
def get_stemmed_text(corpus):
    from nltk.stem.porter import PorterStemmer
    stemmer = PorterStemmer()
     return [' '.join([stemmer.stem(word) for word in review.split()]) for review in corpus]

stemmed_reviews = get_stemmed_text(reviews_train_clean)
```

After steeming the given text:

"thi is not the typic mel brook film it wa much less slapstick than most of hi movi and actual had a plot that wa follow lesli ann warren made the movi she is such a fantast under rate actress there were some moment that could have been flesh out a bit more and some scene that could probabl have been cut to make the room to do so but all in all thi is worth the price to rent and see it the act wa good overal brook himself did a good job without hi characterist speak to directli to the audienc again warren wa the best actor in the movi but fume and sailor both play their part well"

Lemmatization

Lemmatization works by identifying the part-of-speech of a given word and then applying more complex rules to transform the word into its true root.

After processing and lemmatization of the text:

"this is not the typical mel brooks film it was much less slapstick than most of his movies and actually had a plot that was followable leslie ann warren made the movie she is such a fantastic underrated actress there were some moments that could have been fleshed out a bit more and some scenes that could probably have been cut to make the room to do so but all in all this is worth the price to rent and see it the acting was good overall brooks himself did a good job without his characteristic speaking to directly to the audience again warren was the best actor in the movie but fume and sailor both played their parts well"

Tokenization

In order for this data to make sense to our machine learning algorithm, we’ll need to convert each review to a numeric representation, which we call vectorization. The simplest form of this is to create one very large matrix with one column for every unique word in our dataset. Then we transform each review into one row containing 0s and 1s, where 1 means that the word in the corpus corresponding to that column appears in that review. That being said, each row of the matrix will be very sparse (mostly zeros). This process is also known as one-hot encoding.

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(binary=True)
cv.fit(reviews_train_clean)
X = cv.transform(reviews_train_clean)
X_test = cv.transform(reviews_test_clean)
```

Natural Language Processing

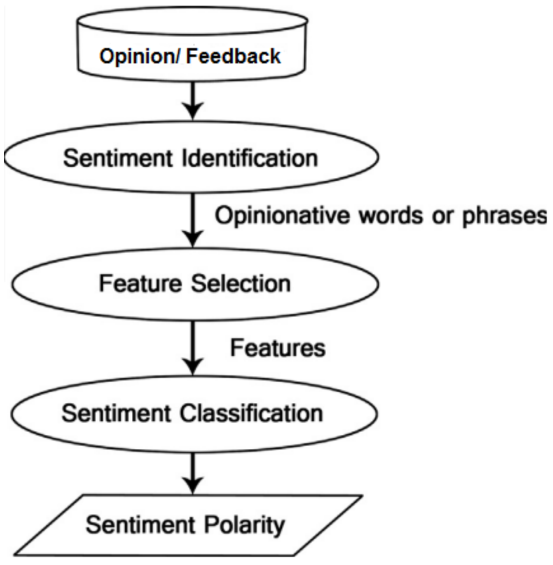
NLP is an interdisciplinary field concerned with the interactions between computers and human natural languages (*e.g:* English) — speech or text. NLP-powered software helps us in our daily lives in various ways, for example:

Personal assistants: Siri, Cortana, and Google Assistant.
Auto-complete: In search engines (*e.g:* Google, Bing).
Spell checking: Almost everywhere, in your browser, your IDE (*e.g:* Visual Studio), desktop apps (*e.g:* Microsoft Word).
Machine Translation: Google Translate.

NLP is short for Natural Language Processing. As you probably know, computers are not as great at understanding words as they are numbers. This is all changing though as advances in NLP are happening every day. The fact that devices like Apple's Siri and Amazon's Alexa can (usually) comprehend when we ask the weather, for directions, or to play a certain genre of music are all examples of NLP. The spam filter in your email and the spellcheck you've used since you learned to type in elementary school are some other basic examples of when your computer is understanding language.

Sentiment Analysis Model

Sentiment Analysis can be considered a classification process as illustrated in Fig. 1. There are three main classification levels in SA: document-level, sentence-level, and aspect-level SA. Document-level SA aim to classify an opinion document as expressing a positive or negative opinion or sentiment. It considers the whole document a basic information unit (talking about one topic). Sentence-level SA aims to classify sentiment expressed in each sentence. The first step is to identify whether the sentence is subjective or objective. If the sentence is subjective, Sentence-level SA will determine whether the sentence expresses positive or negative opinions.



Sentiment Analysis task is considered a sentiment classification problem. The first step in the SC problem is to extract and select text features.
Some of the current features are:
Terms presence and frequency: These features are individual words or word n-grams and their frequency counts. It either gives the words binary weighting (zero if the word appears, or one if otherwise) or uses term frequency weights to indicate the relative importance of features.
Parts of speech (POS): finding adjectives, as they are important indicators of opinions.
Opinion words and phrases: these are words commonly used to express opinions including good or bad, like or hate. On the other hand, some phrases express opinions without using opinion words. For example: cost me an arm and a leg.
Negations: the appearance of negative words may change the opinion orientation like not good is equivalent to bad.
There are many methods and algorithms to implement sentiment analysis systems, which can be classified as:

- Rule-based systems that perform sentiment analysis based on a set of manually crafted rules.
- Automatic systems that rely on machine learning techniques to learn from data.
- Hybrid systems that combine both rule-based and automatic approaches.

Rule-based Approaches

Usually, rule-based approaches define a set of rules in some kind of scripting language that identify subjectivity, polarity, or the subject of opinion.

The rules may use a variety of inputs, such as the following:

Classic NLP techniques like stemming, tokenization, part of speech tagging and parsing.
Other resources, such as lexicons (i.e. lists of words and expressions).

Automatic Approaches

Automatic methods, contrary to rule-based systems, don't rely on manually crafted rules, but on machine learning techniques. The sentiment analysis task is usually modeled as a classification problem where a classifier is fed with a text and returns the corresponding category, e.g. positive, negative, or neutral (in case of polarity analysis is being performed).

Hybrid Approaches

The concept of hybrid methods is very intuitive: just combine the best of both worlds, the rule-based and the automatic ones. Usually, by combining both approaches, the methods can improve accuracy and precision.

The Training and Prediction Processes

In the training process, the model learns to associate a particular input (i.e. a text) to the corresponding output (tag) based on the test samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and tags (e.g. positive, negative, or neutral) are fed into the machine learning algorithm to generate a model.

In the prediction process, the feature extractor is used to transform unseen text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted tags (again, positive, negative, or neutral).

Model for training and testing:

A big part of machine learning is classification — we want to know what class (a.k.a. group) an observation belongs to. The ability to precisely classify observations is extremely valuable for various business applications like predicting whether a particular user will buy a product or forecasting whether a given loan will default or not. The proposed system extensively uses sentiment analysis model and for analyzing the sentiment, this project uses and compares the efficiency for 2 widely used algorithms namely logistic regression and random forest.

1. **Logistic Regression**

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y, can take only discrete values for a given set of features(or inputs), X.

Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.

The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall to be 1, but this seldom is the case.

Some of the advantages of the logistic regression algorithm are:

- Don’t need to pick learning rate
- Often run faster (not always the case)
- Can numerically approximate gradient for you (doesn’t always work out well)

2. **Random Forest**

Random forest is a supervised learning algorithm that is used for both classifications as well as regression. However, it is mainly used for classification problems. As we know, a forest is made up of trees and more trees mean more robust forests. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method that is better than a single decision tree because it reduces the over-fitting by averaging the result. Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science-speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for a random forest to perform well are:

- There needs to be some actual signal in our features so that models built using those features do better than random guessing.
- The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Module Description

The whole project revolves around the collection of feedback processing and calculating the required score. For the above process, there are 3 different modules, namely student module for collection of the feedback, the sentiment analyzer model to calculate the score and store it in the database and faculty module to display the score for the faculty. the 3 modules are explained in detail below.

- **Student Module**

The module is mainly designed for the student where the student can enter their feedback. The student can use this module over a website or an app to login himself/herself. Once logged in the student sees the dashboard of the module where he is shown all the subjects he/she is registered to along with the faculty name he/she is registered to as a subject can be thought of by many different faculty members. Every day after the lecture session is over, the student can log in to the system and post the feedback in the module. As explained above, the student needs to post the feedback in a textual format and not in the form of a score or stars.

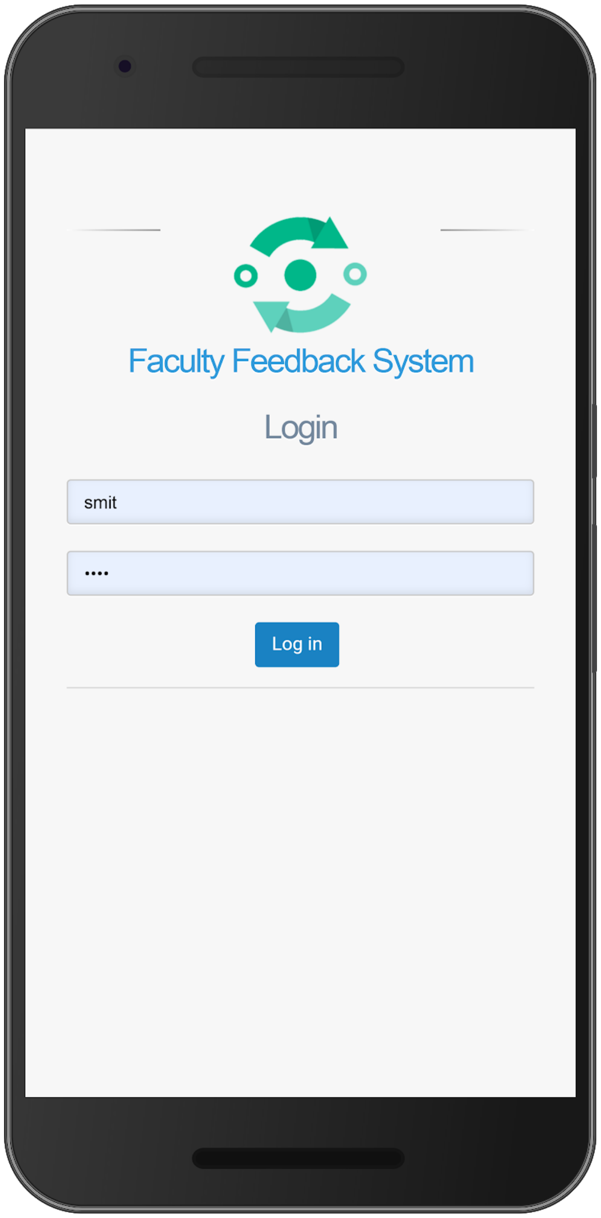
- **Sentiment Analysis Module**

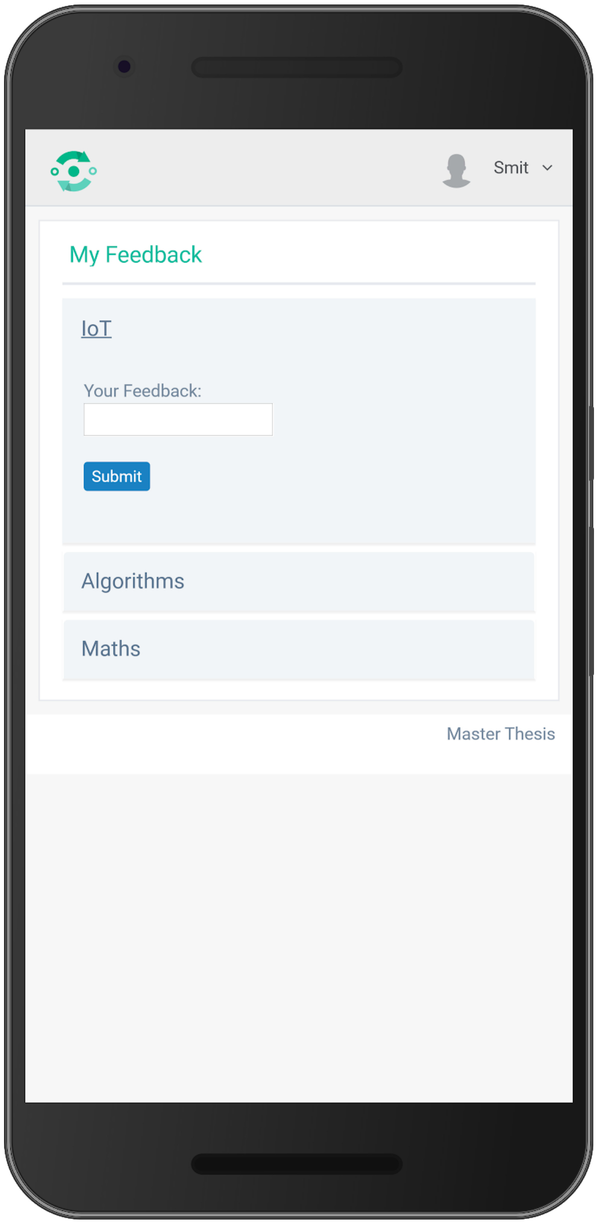
Once the student posts his / her feedback in the textual format, the feedback text is then given to the sentiment analyzer model to perform the sentiment analysis on the same. The sentiment analysis is performed and a score is calculated in the form of positive, negative and neutral. Once the score is calculated, before storing in the database the average score is calculated and the new calculation is added to the database. This will calculate the overall feedback score for the faculty.

- **Faculty Module**

Once the above process of collecting the feedback, calculating the score, and storing the average score in the database is done, the score can be seen in the faculty module. The faculty can see his/her score over the application/ website and according to the score he/she can analyze the lecture session he/she delivered and if required he/she can improve his/her performance which will benefit the students for upcoming sessions.

Screenshots:



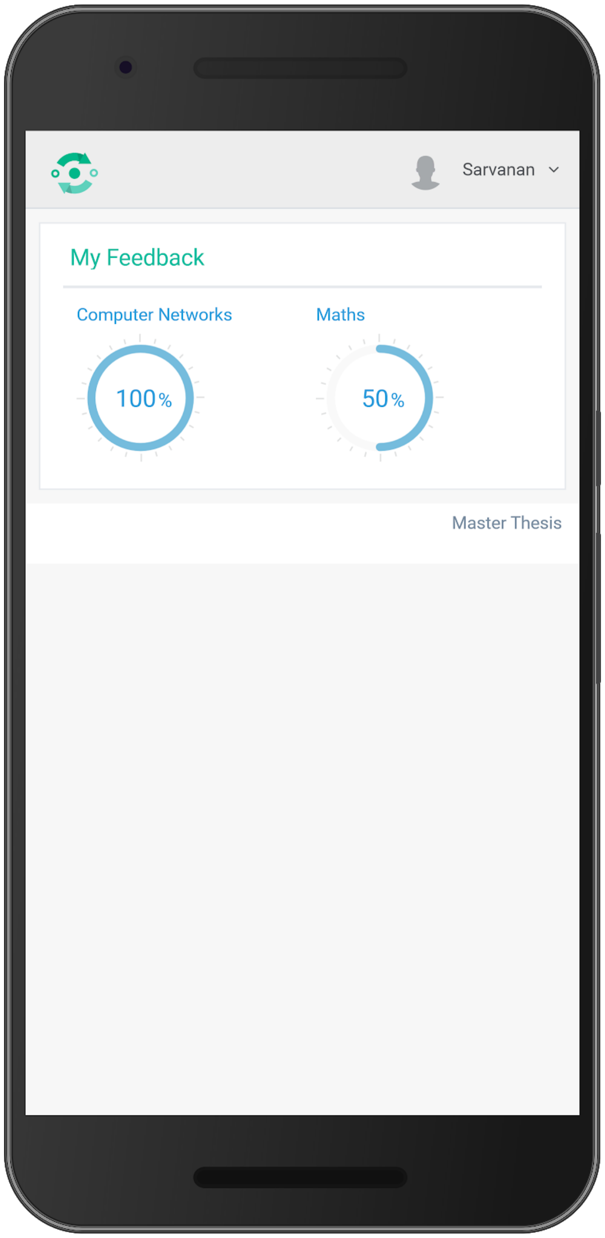


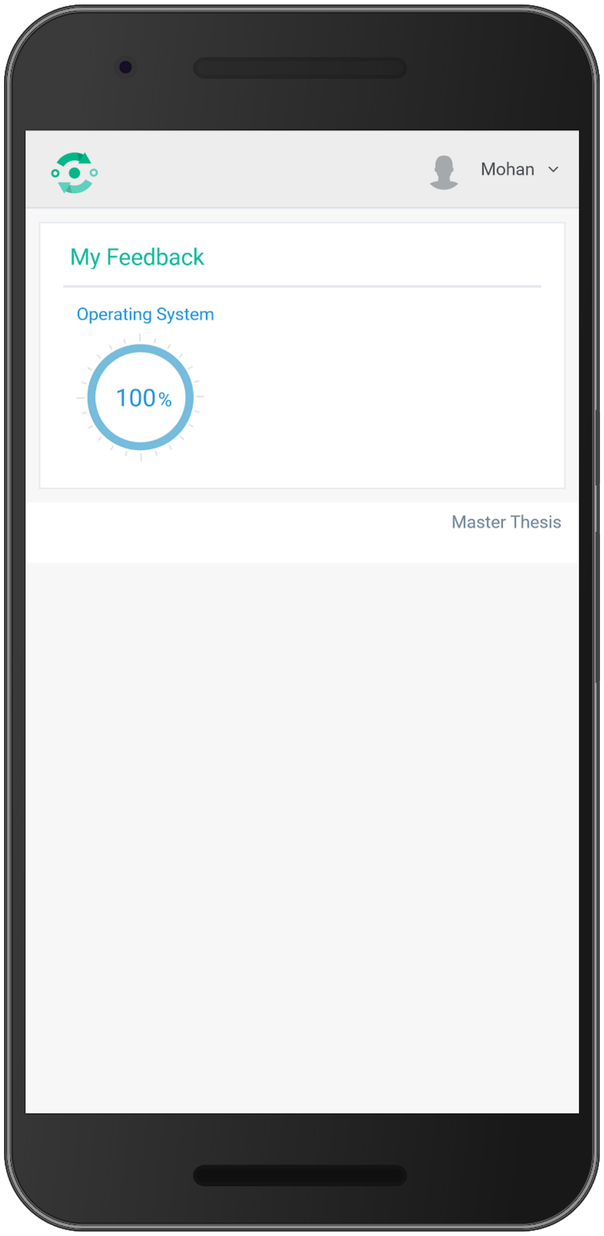
Login screen for faculty/ Student

Feedback Submission Screen

This screen is the login page for the app.

Here the list of subjects will appear for which the student has registered, later from this screen the student can submit his/ her feedback to the respective subject as well as faculty.





Feedback response screen

The feedback given by the student to the faculty goes to the SA Model and after calculating the score: 1 for positive, 0 for negative and 0.5 for neutral calculates the cumulative score and inserts the value to the corresponding faculty's record. The faculty can log in to his/her app/module and check the score.

References:

[1] M. Rathi, A. Malik, D. Varshney, R. Sharma and S. Mendiratta, "Sentiment Analysis of Tweets Using Machine Learning Approach," 2018 Eleventh International Conference on Contemporary Computing (IC3), Noida, 2018, pp. 1-3.

[2] R. Mehra, M. K. Bedi, G. Singh, R. Arora, T. Bala and S. Saxena, "Sentimental analysis using fuzzy and naive bayes," 2017 International Conference on Computing Methodologies and Communication (ICCMC), Erode, 2017, pp. 945-950.

[3] R. A. Kassim and N. Buniyamin, "Evaluating teaching quality using data from student online feedback system," 2015 IEEE 7th International Conference on Engineering Education (ICEED), Kanazawa, 2015, pp. 64-68.

[4] S. Shaikh, P. Sonawane, R. Patil and R. Neeraj, "Knowledge extraction from online feedback system using ontology," 2015 International Conference on Technologies for Sustainable Development (ICTSD), Mumbai, 2015, pp. 1-5.

[5] M. Fernandez-Gavilanes, T. ´ Alvarez-L ´ opez, J. Juncal-Mart ´ ´ınez, E. Costa-Montenegro, and F. J. Gonzalez-Casta ´ no, “Unsupervised ˜ method for sentiment analysis in online texts,” Expert Systems with Applications, vol. 58, pp. 57–75, 2016.

[6] Abdul Aziz, Azlan, Mohd Ali Mohd Isa (2010). SuFO Guidelines (For students, Instructors And Sufo Administrator) University Teknologi MARA. University Teknologi MARA, iLearn.

[7] A. Go, R. Bhayani, and L. Huang. (2009) Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford. [Online]. Available: http://s3.eddieoz.com/docs/sentiment analysis/Twitter Sentiment Classification using Distant Supervision.pdf

[8] C. Troussas, M. Virvou, K. Junshean Espinosa, K. Llaguno, and J. Caro, “Sentiment analysis of Facebook statuses using naive bayes classifier for language learning,” Information, Intelligence, Systems and Applications, vol. 4, pp. 1–6, 2013.

[9] A. Ortigosa, J. M. Martin, and R. M. Carro, “Sentiment analysis in Facebook and its application to e-learning,” Computers in Human Behavior, vol. 31, pp. 527 – 541, 2014.

[10] J. Martin, A. Ortigosa, and R. Carro, “Sentbuk: Sentiment analysis for e-learning environments,” International Symposium on Computers in Education (SIIE), vol. 12, pp. 212–217, Oct 2012.

[11] W. Wang and J. Wu, “Emotion recognition based on cso&svm in e-learning,” International Conference on Natural Computation (ICNC), vol. 7, pp. 566–570, 2011.

[12] Annie McKillop, Robyn Ramage (2005). Time after time: getting some zip back into your info lit. LIANZA Conference,. Christchurch, NZ: 114.

[13] N. Altrabsheh, M. Cocea, and S. Fallahkhair, “Learning sentiment from students feedback for real-time interventions in classrooms,” in Adaptive and Intelligent Systems. Springer, 2014, pp. 40–49.

[14] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision.Technical report, Stanford.

[15] Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of LREC