Prediction Models for Automatic Assessment to Students' Freely-written Comments

Jihed Makhlouf and Tsunenori Mine
Kyushu University, Fukuoka, Japan

Keywords: Educational Data Mining, Natural Language Processing, Genetic Programming, Comments Mining.

Abstract: Tracking students' learning situations is taking a fundamental place in educational institutions. Thanks to the advances in educational technology, we are able to gather more and more data about students using educational software systems. Analyzing such data helped researchers build models that could predict students' behaviors and scores. However, in classroom-based settings, teachers and professors find difficulties to perfectly grasp all their students' learning attitudes. In an approach to address this issue, we asked the students to give freely-written comments answering predefined questions about their learning experience. Thereafter, professors read these comments and give feedback to each student. Nonetheless, professors find themselves overwhelmed by the number of comments which make this approach not scalable to multiple classes for the same professor. In this paper, we address this issue by building a model that can automatically assess the students' comments. We use two different approaches. In the first approach, we treat all student comments the same way, regardless of which question they are related to. The second approach consists of building different individual models that analyze students' comments depending on the question. Experimental results show that the prediction accuracy of assessment to student comments can reach 74%.

1 INTRODUCTION

Thanks to the progress made in educational technology, more and more institutions are incorporating educational software systems. In fact, the usage of such systems allows institutions to gather valuable data which help them improve not only the learning environment but also the students' success and retention (Siemens and Long, 2011; Macfadyen and Dawson, 2010; Dietz and Hurn, 2013). Moreover, students' performance is a topic of interest to many researchers. Many of them built predictive models of students' performances. Using the results of these predictive models, instructors can take actionable decisions to improve the learning experience of students, especially for low-performing students. Nonetheless, building predictive models of students' performances require methods of assessing students' performances in the first place. Countless smart and novel solutions were created seeking to enhance learners' performances. However, the assessment of students' performances has to be considered as a continuous process which, ultimately, aims to increase the quality of student's learning (Hume and Coll, 2009).

There are many ways to assess the learner's performance. Various regular assessment methods like questionnaires and test scores are being used in different educational settings (Minami and Ohura, 2014; Minami and Ohura, 2015). However, in classrooms, teachers do not only teach students, but also try to acquire any information about the students' learning experience through careful observation of individual behaviors and reactions toward lessons. Giving immediate feedback to the class is an effective way to improve learning attitudes (Goda and Mine, 2011). Moreover, in the classroom, there are different types of students based on their rigor, understanding and the overall learning experience. That is what makes it hard for instructors to fully grasp all the student's learning attitudes during the whole semester or teaching period.

Using the regular data sources such as test scores and attendance records does help but is not enough to comprehend and interpret the full spectrum of learning attitudes of all students (Flórez and Sammons, 2013; Yamtim and Wongwanich, 2014). Many institutions use questionnaires to get direct feedback from the students. However, few researches were conducted by using the data from these questionnaires to

Makhlouf, J. and Mine, T.
Prediction Models for Automatic Assessment to Students' Freely-written Comments.
DOI: 10.5220/0009580300770086
In *Proceedings of the 12th International Conference on Computer Supported Education (CSEDU 2020) - Volume 1*, pages 77-86
ISBN: 978-989-758-417-6

build prediction models of students' learning experience.

In this paper, we analyze the freely-written comments made by students after each lesson. These students are asked to answer 5 predefined questions related to several aspects of their learning experience following the PCN method (Goda and Mine, 2011). Afterward, the professors read these comments and reply to them individually. However, this task becomes tedious as the number of students increases. Therefore, the objective of this work is to automate the process of assessing students' comments by building predictive models. We also investigate two different approaches to build the models by taking into account the context of the questions asked to students. The broader outcome of this work is lifting the professors' need to read a large number of comments to get an idea about the students' learning experience. This objective can be achieved by using the predictive models that can predict the students' learning experience by using their comments. On the other side, professors receive an immediate approximation of each student's learning experience. Therefore, professors can use these predictions for better planning of the next lectures according to the students' learning experience, without the necessity to read a large number of comments. Empirical results of this work show that the prediction models are somehow reliable for this task since they can achieve an accuracy score of 74%.

The rest of the paper is organized as follows. In Section 2, we discuss the related work and elaborate on some shortcomings of previous researches. Section 3 is dedicated to the methodology followed to achieve our goal. In Section 4, we provide the results of the conducted experiments. Afterward, in Section 5, we discuss the obtained results. Finally, in Section 6, we present our conclusions and future work.

2 RELATED WORK

The main objective of the educational institutions is to provide a quality education through improving the learning experience of the students. Tremendous efforts are being made to fulfill this objective. This produced a large quantity of high-level research and system designs for educational software. Topics such as predicting students' scores, dropout rates, and even behaviors and affect are more and more popular. Even though questionnaires and surveys are being used for a long time, thorough research using solely data from these questionnaires is still limited in the recent years.

For example, (Jiang et al., 2016) used a very large set of undergraduate course evaluations and built a linear regression model to detect the factors that affect the course and teacher appraisals. (Bachtiar et al., 2011) proposed a questionnaire that quantifies students' affective factors such as motivation, attitude, and personality, then they predicted the students' English language ability considering reading, speaking and writing separately. However, the usage of textual input in the questionnaire is even more limited. A handful of researchers were interested in using textual data from questionnaires. For instance, (Minami and Ohura, 2013) extracted textual input from the term-end questionnaires, combined it with other data, such as exam scores, attendance and homework score and detected common writing characteristics of highly successful students. (Sliusarenko et al., 2013) took the textual data from open-ended comments provided by students while taking a course evaluation survey. The authors used these comments to investigate the most important points from the students' comments and how they are related to their rating of a course. This questionnaire was aimed at providing a rating of the course.

Furthermore, other studies used questionnaires to ask students to self-reflect on their learning experience. In fact, (Goda and Mine, 2011) proposed the PCN method. In this method, the authors asked students to describe their learning experience using freely-written comments after each lesson. PCN is an abbreviation of Previous, Current and Next. The PCN method allows teachers to acquire temporal information about the learning status of each student in a set of (P, C, N). The P (Previous) subset is related to the previous activities that the student did to prepare for the current class or efforts done to review the previous class. In the C (Current) subset, each student describes his/her learning experience, understanding, and achievement during the actual class time. Finally, N (Next) covers students' comments about their plans to review and prepare for the next class. According to the authors, the PCN method had a positive impact on pushing students to have a better self-reflection about their learning strategies; it allowed teachers to gather more knowledge about the tendencies in students' learning settings.
indent Following the implementation of the PCN method, many studies tried to predict the students' performances using their comments from the PCN method. Shaymaa et al. (Sorour et al., 2014a) used clustering with Latent Semantic Analysis (LSA) to predict students' scores. Later, they used Artificial Neural Networks with overlapping methods (Sorour et al., 2014b; Sorour et al., 2015c). In a different

Table 1: Questions and comments following the PCN method.

Subset	Question	Example of comments
P	What did you do to prepare for this lecture?	I read the syllabus.
C	Do you have anything you did not understand? Any questions?	I had problems installing and running the environment.
	What are your findings in this lesson?	I understood the basics of functional programming.
	Did you discuss or cooperate with your friends?	I talked with my friends about errors in my computer.
N	What is your plan to do for the next lecture?	I will do my best to avoid my errors and submit the report.

approach, they used topic modeling (Sorour et al., 2015b) and evaluated students' comment as a time-series problem (Sorour et al., 2015a).

More recently, the same authors built performance prediction models using majority vote while taking into account the succession of lessons when analyzing the students' comments (Sorour et al., 2017).

Nonetheless, gathering students' comments using the PCN method has its own shortcomings. In fact, it requires professors to read a large number of comments which can increase if the same professor has many different classes/students. Moreover, professors have to give individual feedback to each comment which makes it even more time-consuming. Therefore, this method is not effective in grasping the students' learning experience and in a scalable way. In fact, the professors ask students to provide their comments in a dedicated Moodle questionnaire or using a web application accessible by the authorized students. Then, the professors have to read these comments after each lesson to get a broader idea of the learning experience of the students.

Therefore, the aim of this research is to address these shortcomings by building predictive models that can assess the students' freely-written comments. These models will infer the students' learning experience and report them to the professor. The professor can use the aggregation the all students' predicted learning experience to address the problems faced during the lesson, when planning for next lesson.

3 METHODOLOGY

3.1 Data Acquisition

The gathered dataset consists of multiple comment files. Each file represents a lesson. We have an overall of 7 lessons, each lesson is 3 hours long, and all of them are part of a Functional Programming course. 54 Japanese students enrolled in this course during the spring semester of 2017. Each row in the data files represents a "response" of a student in a particular lesson. Each "response" contains 5 comments answering 5 questions following the PCN method. Details about these questions are presented in Table 1. In fact, as shown in Table 1, the subset P (Previous) describes students' activities prior to the lesson. The subset C (Current) contains three different questions. First, students can specify which parts of the lesson they did not understand or had trouble with. In the second question, students outline which new findings they did retain during this lesson. Finally, in the third question, we ask students to report their teamwork and collaboration with classmates during the lesson. The last subset N (Next) is related to students' plans ahead of the next lesson. Some students were absent in some lessons, while others forgot to write their "responses" in several lessons. Because of this, we only had stored 329 students' "responses" out of a maximum of 378 "responses". Overall, we gathered roughly 1645 individual comments, all questions included.

3.2 Data Annotation

The main objective is to automate the assessment of the students' comments, therefore the first step is to manually annotate the dataset following a predefined grid of scores. Table 2 lists the scores and their respective meanings and attributes. In fact, scores are between 1 to 5. The higher the score, the better the comment and the relative learning experience expressed by the student. Due to the relative simplicity of the task, the manual annotation was carried out by two students in the Master program. A deep understanding of the topic of the course was not required. Nonetheless, the dataset is gathered from the Functional Programming course for undergraduates. Moreover, the annotators were not requested to judge the understanding of the students, rather they were asked to rate the quality of the comment and the relative learning experience expressed by the student who made it. Once the annotation phase finished, for each comment, we take the round of the average value of both scores.

3.3 Initial Data Exploration

When gathering the data, we did notice that some lessons did not have comments from the students. This can be explained by some absence of some students or because they did forget to report their activity for a few lessons. As shown in Figure 1, only the 6th lesson has comments from all the students, while the 4th lesson has the least number of students sending their comments (42).

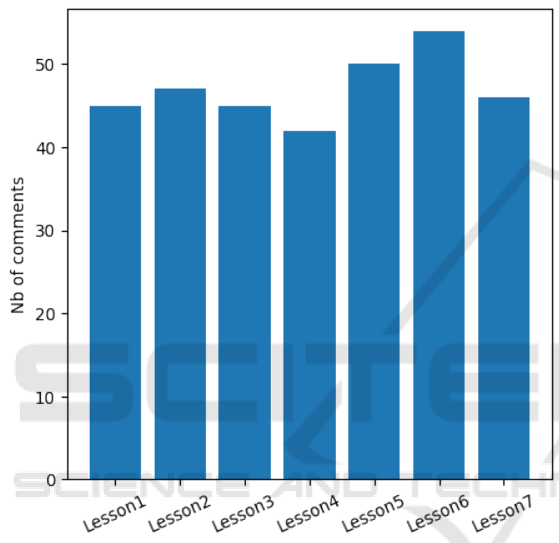

Figure 1: Number of comments per lesson.

The first step is to clean the dataset from missing and inconsistent data. We also discard duplicated comments that answer exactly the same way to the same question. After this clean-up, we retrieve 1158 individual comments which constitute the final dataset. These comments are corresponding to 5 predefined different questions. Therefore, the answers are also different. Students might be more expressive when answering a question compared to another one. Accordingly, we investigate how long are the students' comments depending on the type of the question. Figure 2 shows the distribution of comments' length depending on the question type. From the figure, we can see that P and N comments have a somehow similar distribution, except that students tend to write more about their next plans than about their preparations. Comments that describe students' problems and findings have also similar distribution and their lengths are more spread than the other comments. Nevertheless, the median length of comments

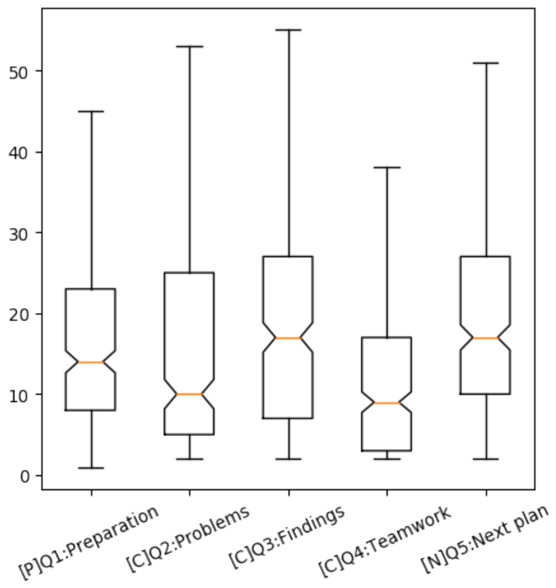

Figure 2: Comment length per Question.

on problems is lower (10) than the median length of comments on findings (17). Meanwhile, teamwork comments are shorter in general than the others.

3.4 Models' Building Approaches

In this research, we explore two different approaches for building our comment-assessment models. The first approach is to build a generalized model in which we mix all comments regardless of the question type.

The second approach is to define 4 separate models, each of them analyzing a specific set of comments. The first model will use the P and N comments. The second model will utilize comments answering the teamwork question. Comments describing findings also have a separate model from the comments about problems and misunderstanding. Even though they are similar somehow, the way the questions are asked and how the answers are formulated gives many comments that are written in the same way but are completely opposite. For example, a comment like ”Nothing”, when it is expressed answering the question ”What you did not understand?” has a positive connotation even if the quality of the comment is poor. When the same comment ”Nothing” is written to reply to the question ”What are your findings in this lesson”, it has rather a negative connotation compared to the previous example. Therefore, comments about findings and comments describing misunderstanding and problems have separated models in this approach.

Table 2: Scores and their meanings.

Score	Meaning
1	No description of the learning actions, or expressions showing a lack of commitment, giving up or negative attitude.
2	Small description of the learning activity without details that make easy to understand the problem or the effort made by the student.
3	Comments that describe briefly with some level of attention to detail the learning activity or showing a moderate degree of commitment, results or troubles.
4	Students expressing their learning activity in details and have a good level of achievement compared to the expectations at that level of the course.
5	Students that achieved the expected level of commitment or practice and who successfully described their learning experience.

3.5 Feature Preprocessing

Before proceeding to the feature transformation, the textual data have to be cleaned. The main language used is Japanese. Nevertheless, English text exists within the comments. Also, since the course is related to programming, students used punctuation signs and special characters in their comments. Therefore, the preprocessing step consisted of cleaning the punctuation and special characters, after that lower case transformation is applied to all English texts used within the comments. Finally, we use MeCab to extract the words and their parts of speech (POS). MeCab[1] is a dictionary-based Part-of-Speech and Morphological Analyzer of the Japanese language.

3.6 Feature Engineering

Aside from the two model-building approaches, we investigate three ways of transforming the textual data. In order to convert the textual data into numerical features, we can use different techniques such as the TF-IDF matrices, the Doc2Vec method, or the pre-trained word embedding.

3.6.1 TF-IDF Matrices

TF-IDF is an abbreviation for Term Frequency - Inverse Document Frequency. Term frequencies are the counts of each word in a document and Inverse Document Frequency is obtained by dividing the total number of documents by the number of documents that contain the word. The Inverse Document Frequency was firstly proposed by Karen Sparck Jones in 1972 in a paper called "A statistical interpretation of term specificity and its application in retrieval" (Sparck Jones, 1972). At that time, it was called the Term Specificity. It is based on counting the number of documents in the collection being searched which contain (or are indexed by) the term in question. The TF-IDF weighting is widely used in information retrieval, text mining, classification, and ranking documents' relevancy. In our case, we will use the TF-IDF weighting vectors as features for our classification model to predict the comment score. Generating the TF-IDF weight matrix was done using the scikit-learn Python machine learning library (Pedregosa et al., 2011).

[1]https://taku910.github.io/mecab/

3.6.2 Doc2Vec

Doc2Vec (Le and Mikolov, 2014) is an unsupervised machine learning algorithm that generates vector representations for paragraphs from pieces of texts. It was inspired by the famous algorithm Word2Vec (Mikolov et al., 2013) that generates word vectors from texts. One of the advantages of Doc2Vec is overcoming the limitations of having a fixed-size input text. Generating the Doc2Vec sentences' representations was accomplished using the gensim Python library (Řehůřek and Sojka, 2010).

3.6.3 Pre-trained Word Embedding

Word embedding is a vector representation of a document's vocabulary. It can grasp the relationship between words and their context. Word2Vec is a famous method of generating this word embedding. However, pre-trained word embedding means that the weights and the vector representation are already generated by training the model on a large corpus of documents. Thanks to fastText, the Facebook C++/Python library (Joulin et al., 2016), we can acquire such a word vector for the Japanese language. In fact, Facebook released pre-trained models for 157 languages, trained on Common Crawl[2] and Wikipedia[3] texts.

[2]https://commoncrawl.org/
[3]https://www.wikipedia.org/

3.7 Workflow

As we defined earlier, we investigate two different approaches to building the assessment models. For each approach, we follow three different ways of feature engineering. Figure 3 shows the whole workflow.

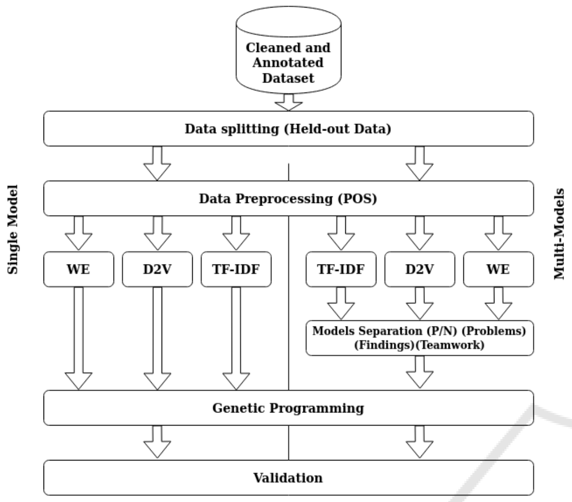
Cleaned and Annotated Dataset
Data splitting (Held-out Data)
Data Preprocessing (POS)
Single Model
Multi-Models
WE
D2V
TF-IDF
TF-IDF
D2V
WE
Models Separation (P/N) (Problems) (Findings)(Teamwork)
Genetic Programming
Validation

Figure 3: Models building workflow.

The first step is to split the data into testing and training. In fact, as it is custom in prediction models, we hold out part of the dataset as unseen validation data. For the Multi-model approaches, we held out comments from each corresponding question (e.g. holding out P or N comments from the P/N model). For the single model, in which all comments are mixed regardless of the type of the question, we held out comments using a stratisfied split. The stratisfied split allows us to respect the proportions of each type of comment in the whole dataset. We used a ratio of $1/4$ of the dataset for unseen validation only data. Then, we use MeCab to extract words and their Part-of-Speech. After that, we proceed to build our models. For each approach, we use the three discussed methods for feature engineering. Therefore, we have 6 different alternatives for comparison. However, in the multi-models approach we create 4 different models and use the appropriate comments for training: Previous and Next comments, Problems comments, Findings comments, and Teamwork comments. When we evaluate each alternative in the multi-models approach, we take the average of the 4 models' performances using equation (1):

$$P_{multi} = \frac{\sum_1^4 PM_i}{4} \tag{1}$$

Here $multi$ is the multi-model alternative and PM_i is the i^{th} model of the respective alternative.

Afterward, each of the 6 alternatives will be optimized separately from the others. In fact, since we compare many different alternatives independently, we want to find the most adequate machine learning method with the best hyper-parameters for each alternative. In order to find this best "pipeline", we use genetic programming as our tool for searching. We do not compare two machine learning methods, but rather we try to give each alternative its best try. Once the optimization phase finished, we validate each alternative "pipeline" and compare their results using the held-out data. Overall, we have 6 alternatives that we explore. Table 3 provides a summary of the name of each alternative and their differences.

3.8 Optimization and Genetic Programming

Briefly, genetic programming is a technique derived from genetic algorithms in which instructions are encoded into a population of genes. The goal is to evolve this population using genetic algorithm operators to constantly update the population until a predefined condition is met. The most common ways of updating the population are to use two famous genetic operators called crossover and mutation. Crossover is used to diversify the research in the research space by taking some parts of the parent individuals and mixing them into the offspring. On the other hand, mutation is the process of updating only some parts of an individual and is used to maintain the actual diversity, in other words, intensify the research in a certain area of the research space. The population is evolving from one generation to another while keeping the fittest individuals in regard to one or many objectives. When using genetic programming for machine learning optimization, each individual represents a pipeline that holds a machine learning technique with its hyper-parameters. We use the pipeline score as the objective function; the pipeline accuracy score is an example of an objective function that has to be maximized.

In our case, we used genetic programming by searching through a multitude of machine learning techniques and their respective hyper-parameters to find out which combination gives the best results. To achieve our goals we used the python library TPOT (Olson et al., 2016). However, in order to use genetic programming, there are several hyper-parameters that we need to initialize.

Table 4 explores the principal hyper-parameters that we have to initialize. The generation count is the number of iterations of the whole optimization process. A bigger number gives better results, but also takes more time to finish. We can also fix a maximum

Table 3: Summary and names of the investigated alternatives.

Name	Characteristics
single_tf-idf	Analyze all comments regardless of the type of the question and generate features using the TF-IDF method.
single_doc2vec	Analyze all comments regardless of the type of the question and generate features using Doc2Vec sentence vectors.
single_pre-trained	Analyze all comments regardless of the type of the question and generate features using pre-trained Japanese language word vectors.
multi_tf-idf	Generate 4 models (P and N; Misunderstanding; Findings and Teamwork) and analyze the comments relative to each model using TF-IDF.
multi_doc2vec	Generate 4 models (P and N; Misunderstanding; Findings and Teamwork) and analyze the comments relative to each model using Doc2Vec.
multi_pre-trained	Generate 4 models (P and N; Misunderstanding; Findings and Teamwork) and analyze the comments relative to each model using pre-trained Japanese word vectors.

amount of time for the whole process. The population size is the number of individuals which will evolve in each iteration, each member of the population represents a machine learning pipeline. The offspring size is the number of individuals that are supposed to be generated from the previous population using the genetic algorithm operators. After executing the operators and generating the offspring, the individuals from the population and the offspring compete to survive and be part of the next population. When the individuals compete against each other, we only keep the fittest ones, meaning the individuals with the best score. The method used to measure the score is defined in the scoring hyper-parameters. We used the accuracy as our scoring method. That means we only keep the individuals (thus the pipelines) which have the highest accuracy values. Mutation and Crossover rates are the probabilities of having respectively a Mutation or a Crossover operation to evolve one or more individuals. We set them to be a 90% chance of having a mutation against a 10% of having a crossover operation. Finally, the TPOT tool gives us the possibility to cross-validate our pipelines internally, therefore we set the number of folds to 5.

Table 4: Genetic Programming hyper-parameters.

Hyper-parameter	Value
Generations count	100
Population size	100
Offspring size	100
Scoring	Accuracy
Mutation rate	0.9
Crossover rate	0.1
Internal Cross Validation	5 folds

Similar results can also be achieved using a grid search. However, it is computationally expensive since a grid search tries all possible combinations of hyper-parameters. On the other hand, genetic programming can be considered as a heuristic-based grid search since it updates the population in an informed way. Therefore, it does not evaluate the combinations that are less likely to give good results. This allows it to be faster than a simple grid search.

4 EXPERIMENTAL RESULTS

We have six alternatives which we optimized to find out which machine learning pipeline fits best for each of them. Table 5 shows the results of the optimization phase, with the chosen machine learning technique and its best score.

Table 5: Results of the optimization process.

Alternatives	Best Method	Best Score
single_tf-idf	Random Forest Classifier	0.633
single_doc2vec	Random Forest Classifier	0.619
single_pre-trained	Random Forest Classifier	0.676
multi_tf-idf	K-Nearest Neighbors	0.705
multi_doc2vec	P+N: SVM Misunderstand: Random Forest Classifier Findings: K-Nearest Neighbors Teamwork: Random Forest Classifier	0.662
multi_pre-trained	Random Forest Classifier	0.740

In the first approach, in which we mix all comments regardless of the type of the question, we notice that for all its alternatives, the Random Forest Classifier was the best machine learning method, with the best scores of 0.633, 0.619, 0.676 respectively for the usage of TF-IDF, Doc2Vec and the pre-trained word embedding. For the separated models, we have various machine learning methods, giving the best results. In fact, when using the TF-IDF weighting, all four separated models have K Nearest Neighbor as the best classifier with an average score of 0.705. However, when using the Doc2Vec technique, different machine learning methods give the best results to each separated model. In fact, for the P (Previous) + N (Next) model, Support Vector Machine (SVM) is the best. For the misunderstanding model, the Random Forest Classifier achieves the best score similarly to the teamwork model. Finally, the model of findings uses K Nearest Neighbors. The last alternative, in which we used a pre-trained word vector and four different models, get the Random Forest Classifier as the best performing method for all the models. The average score in this alternative is 0.740.

Once we decided which machine learning methods we will use, we train the models accordingly and proceed to validate each performance using unseen data. In the validation phase, we did measure various metrics to evaluate the performances of our assessment models.

Table 6 shows the validation scores of all models. The best scores are written in boldface. The first notice is that all models did perform better than chance in giving the right score to the comments. The best results in terms of accuracy and precision are achieved by the multi-models approach using the pre-trained word embedding having attained an accuracy of 0.740 and a precision of 0.668. This model also scored second best in the recall with 0.630 and also second-best in the F1-score having 0.635. With the same approach, but using the TF-IDF weighting matrix, we achieve the best scores in recall with 0.660 and in F1-score having 0.650. This model is the second-best in accuracy attaining 0.662 similarly in precision by obtaining 0.655.

Table 6: Validation scores for all models.

	Accu	Prec	Recall	F1-score
single tf-idf	0.603	0.560	0.600	0.560
single doc2vec	0.586	0.510	0.590	0.530
single pre-trained	0.590	0.550	0.590	0.550
multi tf-idf	0.662	0.655	**0.660**	**0.650**
multi doc2vec	0.548	0.545	0.548	0.505
multi pre-trained	**0.740**	**0.668**	0.630	0.635

5 DISCUSSION

We notice that the first approach, in which we gather all comments regardless of the question answered, did not perform better than the four distinguished models. Another notice is that across both approaches, using Doc2Vec for building the features set performed the worse, beaten by the use of TF-IDF and the pre-trained word embedding in each metric. We also observe that using the pre-trained word embedding contributes to a boost in performance for the separated models approach, but not in the single model approach in which we include all comments regardless of the question being answered.

Separating models for each question is the best approach in this dataset, however, there are a few shortcomings in following this approach in the long term. In fact, this approach is not easily maintainable or scalable. First, any improvement or update in this approach has to be replicated as many times as there are separated models. Furthermore, if we are planning to add more questions or implement an interactive interface to gather students' comments, these models cannot respond very well. Nevertheless, in the actual situation and scale, they might be very valuable and effective in assessing students' comments without much human intervention.

On the other hand, building a model capable of generalizing well toward comments regardless of the initial question or aspect is considered to be the right way. Not only in terms of scalability, but also in the complexity of the whole system. One reason that might be behind these under-performing models is the small semantic conflict between the question about misunderstanding and findings. As explained above, the second question asks students to report anything that they did not understand. While the third question asks them to detail any findings they got from the lesson. Answers such as "Nothing" or "Nothing in particular" are viewed differently. Saying "Nothing" to mean "I don't have any problem" is more positive than meaning "I did not have any findings in this lesson". Therefore, mixing such comments under the same model is tricky and relies on the subjective human assessment when manually annotating comments. A possible solution is to find a way to incorporate the information about which aspect is covered by the comment without altering its meaning, intent or semantic weight might provide a noticeable improvement in the models' performance.

In the process of finding the best machine learning pipeline, the usage of genetic programming is a great help. In fact, it can be considered as a heuristic-based grid search. Thus, it covers a big portion of the research space of the possible combination of values. But, in the same time, it reduces a lot of the time to achieve this, since it does not try all possibilities, but rather try the ones that are more likely to give good results because of the mechanism of elitism where only the fittest (having the best score) individuals are kept in the population for the next iteration.

6 CONCLUSION AND FUTURE WORK

Tracking students' learning experience is crucial to understand their behaviors and intervene when it is needed. However, in a classroom-based environment, instructors' careful observations are not enough to grasp the whole spectrum of the learning behaviors of all students during the whole semester. Therefore, tools that provide a way to track students' learning experiences are very helpful. In an approach to overcome many difficulties in the continuous tracking of students' learning environments, (Goda and Mine, 2011) proposed the PCN method, in which students write freely their comments and describe their learning activities in regard to three aspects. P (Previous) mainly describes activities for preparing the lesson. C (Current) is related to the learning activities during the class. N (Next) gathers comments from students describing their plans for the next lesson. Early research achieved success while predicting students' performances by using the PCN method. However, teachers have to spend a lot of time reading comments and giving proper feedback. This paper is an approach to automate the assessment of these students' comments. We proposed two methods to build the automatic assessment model. The first is to gather all comments regardless of which questions they are related to, and treat all comments the same way. The second approach is to build different models depending on the question answered by that comment. Therefore, four models were considered. A model for P and N comments, then three models for C comments. One for comments reporting misunderstanding, a second for students describing their findings during the lesson and lastly a model for comments related to teamwork and group collaboration. For each approach, we try three different ways of building our features set and investigate their impact. The first way is to use the TF-IDF weights, the second is using Doc2Vec and the third way is to load a pre-trained word embedding.

Empirical results showed that the second approach of building separate models dealing with each type of comment is outperforming the first approach of mixing comments. Secondly, we observed a significant improvement in the classification performances when we use the pre-trained word embedding.

Some improvements are considered for this work. Firstly, an investigation of how the first approach will result if we find a way of providing information about which question is answered when all comments are mixed. It might be a sort of padding that will depend on the question type. Another improvement is to annotate comments not only with one score, but with different scores that assess different aspects of the comment, such as how much descriptive is the comment? also, assess the efforts made by the students or the degree of his self-rigor and self-motivation. This sort of annotation will enable a more granular analysis of the students' comments and open the doors for more robust and better performing automatic comments assessment models.

We are also planning to implement this research outcome into a fully functional platform, where professors can monitor the lesson and aggregate the students' learning experience to better plan for the next lecture. Moreover, the results of the prediction models can be used to detect low-performing students so that professors can give them the appropriate guidance to improve their understanding of the course material.

ACKNOWLEDGEMENTS

This work is partially supported by JSPS KAKENHI No.JP16H02926, JP17H01843, JP18K18656 and JP19KK0257.

REFERENCES

Bachtiar, F., Kamei, K., and Cooper, E. (2011). An estimation model of english abilities of students based on their affective factors in learning by neural network. In *proceedings of IFSA and AFSS International Conference 2011*.

Dietz, B. and Hurn, J. (2013). Using learning analytics to predict (and improve) student success: A faculty perspective. *Journal of Interactive Online Learning*, 12:17–26.

Flórez, M. T. and Sammons, P. (2013). *Assessment for Learning: Effects and Impact.* ERIC.

Goda, K. and Mine, T. (2011). Pcn: Qualifying learning activity for assessment based on time-series comments. In *Special Session of The 3rd International Conference on Computer Supported Education: Assessment Tools and Techniques for e-Learning (ATTeL 2011).*

Hume, A. and Coll, R. (2009). Assessment of learning, for learning, and as learning: New zealand case studies. *Assessment in Education: Principles, Policy and Practice*, 16.

Jiang, Y., Syed, S. J., and Golab, L. (2016). Data mining of undergraduate course evaluations. *INFORMATICS IN EDUCATION*, 15:85–102.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Macfadyen, L. and Dawson, S. (2010). Mining lms data to develop an "early warning system" for educators: A proof of concept. *Computers and Education*, 54:588–599.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Minami, T. and Ohura, Y. (2013). Investigation of students' attitudes to lectures with text-analysis of questionnaires. In *Proceedings - 2nd IIAI International Conference on Advanced Applied Informatics, IIAI-AAI 2013*, pages 56–61.

Minami, T. and Ohura, Y. (2014). A correlation analysis of student's attitude and outcome of lectures investigation of keywords in class-evaluation questionnaire. *Advanced Science and Technology Letters*, 73:11–16.

Minami, T. and Ohura, Y. (2015). How student's attitude influences on learning achievement? an analysis of attitude-representing words appearing in looking-back evaluation texts. *International Journal of Database Theory and Application*, 8:129–144.

Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Moore, J. H., et al. (2016). Automating biomedical data science through tree-based pipeline optimization. In *European Conference on the Applications of Evolutionary Computation*, pages 123–137. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Siemens, G. and Long, P. (2011). Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, 5:30–32.

Sliusarenko, T., Clemmensen, L., and Ersbøll, B. (2013). Text mining in students' course evaluations: Relationships between open-ended comments and quantitative scores. *CSEDU 2013 - Proceedings of the 5th International Conference on Computer Supported Education*, pages 564–573.

Sorour, S., Goda, K., and Mine, T. (2015a). Evaluation of effectiveness of time-series comments by using machine learning techniques. *Journal of Information Processing*, 23:784–794.

Sorour, S., Goda, K., and Mine, T. (2015b). Student performance estimation based on topic models considering a range of lessons. In *AIED2015*.

Sorour, S., Goda, K., and Mine, T. (2017). Comment data mining to estimate student performance considering consecutive lessons. *Educational Technology Society*, 20:73–86.

Sorour, S., Mine, T., Goda, K., and Hirokawa, S. (2014a). Prediction of students' grades based on free-style comments data. In *The 13th International Conference on Web-based Learning*, volume LNCS 8613, pages 142–151.

Sorour, S., Mine, T., Goda, K., and Hirokawa, S. (2015c). Predicting students' grades based on free style comments data by artificial neural network. *Proceedings - Frontiers in Education Conference, FIE*, 2015.

Sorour, S., Mine, T., Godaz, K., and Hirokawax, S. (2014b). Comments data mining for evaluating student's performance. *Proceedings - 2014 IIAI 3rd International Conference on Advanced Applied Informatics, IIAI-AAI 2014*, pages 25–30.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its applicationin retrieval. *Journal of Documentation*, 28:11–21.

Yamtim, V. and Wongwanich, S. (2014). A study of classroom assessment literacy of primary school teachers. *Procedia - Social and Behavioral Sciences*, 116:2998–3004.