Question classification according to Bloom's revised taxonomy

Joe Harrison
Olivier Dikken
Dennis van Peer

QUESTION CLASSIFICATION ACCORDING TO BLOOM'S REVISED TAXONOMY

Bachelor End Project Report

in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science
at Delft University of Technology
to be defended publicly on the 6th of June 2017 at 13:30

by

Joe HARRISON, Olivier DIKKEN, Dennis VAN PEER

Supervisors:
Claudia Hauff, TU Delft
Martha Larson, TU Delft
Pim de Haan, FeedbackFruits

An electronic version of this thesis is available at http://repository.tudelft.nl/.

PREFACE

In this report we present the work we have done for our bachelor end project at FeedbackFruits. FeedbackFruits is a company that improves education by building an educational platform for teachers. Over the course of 10 weeks we have built a classifier that classifies questions according to Bloom's revised taxonomy. This classifier can be used by our client FeedbackFruits in their current platform to help teachers better align their courses.

ACKNOWLEDGEMENTS

0.1. ACKNOWLEDGEMENTS

In this section we would like to take the opportunity to thank all the people that helped us with our project. We would like to thank our supervisors Claudia Hauff and Martha Larson from the TU Delft for taking the time and effort to advise us on our project. We would also like to thank our client FeedbackFruits for providing us with a nice work environment, much appreciated lunches and above all good company. We immediately felt very welcome at FeedbackFruits due to the relaxed atmosphere in the office. All employees are always ready to give their input on our project, therefore we would like to thank Esther for helping out with the design of the graphical user interface, Wang Bo for advising us on machine learning matters and sending us interesting papers on the subject, and Ewoud de Kok, the CEO of FeedbackFruits, for our brainstorm sessions. We would like to especially thank Pim de Haan for being our advisor at FeedbackFruits and always asking the right questions to get us on track.

CONTENTS

0.2. ABSTRACT

FeedbackFruits is a company that provides tools for educators to organize their courses. The company is currently working on aiding teachers in aligning course material and assessment. Aligning the two provides students with clear expectations and can lead to an increase in learning [1]. Aligning course material and assessment is usually done by comparing what the students are taught to how students are assessed. When a student is assessed by an exam consisting of questions, the alignment process involves classifying these questions according to the cognitive process categories needed to answer them[2]. This process can be time consuming if an exam contains many questions it, and it can be easy to lose oversight of whether the questions in the assessment are representative of what is taught in the course material. The task of classifying questions into categories that represent the cognitive processes needed to answer them can be facilitated by providing a classification tool. This tool also gives educators insight by displaying a summary of the different question categories present in a set of questions. As part of the solution to the problem of course alignment, FeedbackFruits requested the development of a question classifier which classifies questions according to the cognitive process required to answer them. Bloom's revised taxonomy (subsection 2.1.1) is a taxonomy that categorizes questions and learning objectives into six distinct classes in the cognitive process domain. We propose a software solution that uses machine learning techniques to classify a courses' questions and provides a clear overview of the classes in Bloom's revised taxonomy present in these courses. To achieve this, we built a training set and test set by combining a pre-existing labeled dataset from Anwar Ali Yahya, Addin Osama, et al. [5] and a self labeled dataset of over 1500 samples. We engineered a set of features specific to short text samples and questions. We adopted an experimental approach in selecting the classifier model: we tested several different models throughout the project and picked the best performing models as final step. When looking up Bloom's taxonomy it is often presented with lists of class specific keywords. We replicated a study [3] that makes use of keywords that are indicative of the class in Bloom's taxonomy to set a baseline to compare our model to. We ran our model and the model of the baseline study on the same test set. Our model scored an accuracy of 75% compared to the baseline model which scored an accuracy of 40%.

1

INTRODUCTION

1.1. INTRODUCTION

Alignment of learning objectives, instructional activities and assessment is a highly desirable trait for a course to have as it promotes learning in students [1]. Misalignment can cause a decrease of learning in students [1]. Educators can make use of a taxonomy to classify questions used to assess learning objectives into classes to align their courses. Manually classifying each question can be a time consuming task prone to error [4]. It is also easy to lose oversight of whether the questions in the assessment are representative of what is taught in the course material, when the assessment consists of many questions.

FeedbackFruits is an education company that specializes in higher education. The company promotes learning through their platform which enables teachers to get insights into their students' learning activities, such as assignment completion time. FeedbackFruits would also like to give teachers insight into their own courses. They've asked us to build a classifier that classifies questions into cognitive levels of complexity according to Bloom's revised taxonomy (see subsection 2.1.1 and table 2.1). This classifier will then be used for a multitude of parts of their platform. For example: FeedbackFruits is developing a chatbot (EduBot) on their platform to which teachers could submit their exam and get back a bloom classification for each question, which they can use to align their course. The product we are delivering to our client consists of a classifier with tuned parameters and a dataset with which the classifier can be trained. The software will run as a web service. The client will be able to use, extend or retrain the classifier using an API developed by us.

REFERENCES

[1] Phyllis Blumberg. Maximizing learning through course alignment and experience with different types of knowledge. *Innovative Higher Education*, 34(2):93–103, 2009.

ISSN 1573-1758. doi: 10.1007/s10755-009-9095-2. URL http://dx.doi.org/10.1007/s10755-009-9095-2.

[2] Phyllis Blumberg. Maximizing learning through course alignment and experience with different types of knowledge. *Innovative Higher Education*, 34(2):93–103, 2009.

[3] Wen-Chih Chang and Ming-Shun Chung. Automatic applying bloom's taxonomy to classify and analysis the cognition level of english question items. In *Pervasive Computing (JCPC), 2009 Joint Conferences on*, pages 727–734. IEEE, 2009.

[4] K. A. Osadi, M. G. N. A. S. Fernando, and W. V. Welgama. Ensemble classifier based approach for classification of examination questions into bloom's taxonomy cognitive levels. *International Journal of Computer Applications*, 162(4):1–6, Mar 2017. ISSN 0975-8887. doi: 10.5120/ijca2017913328. URL http://www.ijcaonline.org/archives/volume162/number4/27228-2017913328.

[5] Anwar Ali Yahya, Addin Osama, et al. Automatic classification of questions into bloom's cognitive levels using support vector machines. 2011.

2

Problem Definition

2.1. Problem Definition

In this section we will define the problem at hand and further clarify the scope of this project. Consistency among learning objectives, course material, learning activities and course assessment can increase learning in students [4] [2]. Not all educators may align their courses [4]. In the discussion section of Blumberg's paper on maximizing learning through course alignment [4], Blumberg describes a scenario in which the assessment of a course is not aligned with the learning goals which results in a suboptimal learning in students. Tuning the learning objectives, course material, learning activities and course assessment so that they are aligned can be a daunting task for educators, but is an essential part of course planning [4]. It is easy to lose oversight when aligning courses that are assessed by a large amount of questions. An examination in the form of a set of questions is often used as a final assessment of a student's performance in a course. Therefore we have researched whether there is a better way for educators to approach this task while keeping a good overview.

In this project we propose a solution to this problem. We have developed a classifier that classifies questions according to Bloom's revised taxonomy using machine learning methods. The scope of this project is limited to well formulated questions in text form of roughly university level difficulty posed in American English. The classifier will not take spelling mistakes into account. We will also only train the classifier on standalone questions i.e. questions that can be answered without an introduction, questions that don't need an accompanying figure or text and questions that aren't multiple choice.

2.1.1. Bloom's revised taxonomy

Bloom's taxonomy [3] is a taxonomy that classifies educational learning objectives and questions. The taxonomy classifies questions and learning goals in the cognitive process dimension. The classes of the original taxonomy created by Bloom et al. are: Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. It was developed to

Table 2.1: Classes in Bloom's revised taxonomy and examples

Class	Example
Remembering	Who is the founder of FeedbackFruits?
Understanding	Explain the concept of gravity.
Applying	Calculate the hamming distance of the following string: 01010110
Analyzing	Compare an linear SVM to a Naive Bayes classifier.
Evaluating	Justify the use of force in peace keeping missions of the UN.
Creating	Write a report for your bachelor end project.

provide a way to classify educational goals to help teachers discuss curricular and evaluation problems with greater precision. In 2001 Krathwohl et al. [1] published a revised version of Bloom's taxonomy. The revised version's classes are: Remembering, Understanding, Applying, Analyzing, Evaluating and Creating (see table 2.1). The classes in this version are in verb form instead of noun form and the Creating class is at the top of the taxonomy rather than the Evaluating class. The taxonomy was revised to show how the taxonomy intersects and acts upon different types and levels of knowledge - therefore the knowledge dimension was added and in the process the aforementioned changes were also applied. In this project we further limit the scope by only classifying the cognitive process dimension, because no labeled data of the knowledge dimension is available and it would cost too much time to label questions in this dimension in addition to the cognitive process dimension. Any mention to Bloom's revised taxonomy or classes in the taxonomy will be a reference to the cognitive process dimension. It is worth mentioning that there are other taxonomies available e.g. the SOLO taxonomy. We decided to go with Bloom's revised taxonomy as labeled datasets were available and our client FeedbackFruits requested it.

2.1.2. AMBIGUOUS CLASSES

A dataset of labeled questions is essential in our project: In machine learning a model needs a dataset to train on, and during the phases of feature engineering and hyper parameter tuning a dataset is needed to be able to run and evaluate the model. Labeling data isn't always straightforward (section 7.1). Classifying questions according to which cognitive abilities are required to answer them can sometimes require more information than is present in the question alone, e.g. figures, plots and text. When predicting the class of an exam question the classifier does not know the course material or any other information which is sometimes needed to be able to interpret what the question is asking. When labeling questions we encountered two main types of questions that have ambiguous classes.

First there is the case where one question actually contains several questions, e.g. the question: Name a nation and calculate its GDP, has one remember question "Name a nation", and a question that falls in the Apply category of the taxonomy "Calculate its GDP". When encountering samples that could be placed in several classes we always label the ground truth as the highest class in the taxonomy we use to which the sentence could belong. The reason for doing this is that to answer questions of a higher class in

the taxonomy it is often required to also use the cognitive processes required to answer questions of a lower class (subsection 2.1.1).
Second there are samples that require more information to know to which class they belong e.g. "What is an appropriate threshold alpha for the green value of a pixel to make a distinction between red and orange pixels in the RGB color model?". If this is an exam question, from this sentence alone we cannot know what cognitive process the student taking the exam needs to go through to answer the question as this depends on the course material. If during the course this example was looked into and a specific value for alpha was taught then answering the question requires the student to remember that value. If during the course a formula is given to compute alpha given the available information the question belongs to the apply class. This situation arises often in the samples of our dataset and we again made the decision to assign this type of ambiguous question to the highest class it could belong to. Our classifier does not have any more information about the question than the sentence itself when making a prediction. Therefore when labeling these samples - of which the class is ambiguous due to lack of information about the course material - we assume that the course material does not explicitly give the answer to the question, because otherwise all these samples could be labeled as remembering questions.

REFERENCES

[1] Lorin W Anderson, David R Krathwohl, P Airasian, K Cruikshank, R Mayer, P Pintrich, J Raths, and M Wittrock. A taxonomy for learning, teaching and assessing: A revision of bloom's taxonomy. *New York. Longman Publishing. Artz, AF, & Armour-Thomas, E.(1992). Development of a cognitive-metacognitive framework for protocol analysis of mathematical problem solving in small groups. Cognition and Instruction*, 9(2): 137–175, 2001.

[2] John Biggs. Enhancing teaching through constructive alignment. *Higher education*, 32(3):347–364, 1996.

[3] Benjamin S Bloom et al. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay*, pages 20–24, 1956.

[4] Phyllis Blumberg. Maximizing learning through course alignment and experience with different types of knowledge. *Innovative Higher Education*, 34(2):93–103, 2009. ISSN 1573-1758. doi: 10.1007/s10755-009-9095-2. URL http://dx.doi.org/10.1007/s10755-009-9095-2.

3

Problem Analysis

3.1. Problem Analysis

In this section we will analyze the problem in depth and break it down into it's core components. We will first set a baseline to which we compare our results and then analyze other studies to determine which components are useful.

3.1.1. Classifier

At the heart of this problem lies building a classifier that performs well. The classifier should take a question as input and return one of six classes of Bloom's revised taxonomy, preferably the correct one. Ikonomakis et al. [2] removed stop words from and stemmed their text data and then preprocessed and calculated the term frequency (TF) and inverse document frequency (IDF)[3] to get a vector representation of their text data. However this was intended for entire documents larger that the samples used in our project. Khoo et al. [4] showed in their research that common preprocessing techniques such as stop word removal and stemming can have a detrimental effect on classification accuracy for short sentence classification. Since we know our samples are questions we engineered features specific to our problem in addition to using TF-IDF (see **??**) , for example the presence of a word that begins with "wh-", which is common in questions belonging to the class Remembering.

3.1.2. Setting a baseline

Before we build our classifier we have set a baseline to which we can compare its accuracy. We suspect that certain keywords can be indicative of a certain class in the taxonomy. Chang et al. [1] conducted a study on using common keywords of each class in Bloom's revised taxonomy to classify questions. They used lists of common keywords per class in the taxonomy from 8 different sources. The frequencies of how often a keyword-class pairs appears in the different sources are used as weights to determine the class of questions. For example, if 7 out of 8 lists consider the keyword "Analyze" to belong to the class Analyzing, and 1 out of 8 lists considers the keyword "Analyze" to be of the class Re-

membering, then a question that only contains the keyword "Analyze" would be assigned a score of 7/8 in the class Analyzing and 1/8 in the class Remembering. Considering that in this example, "Analyze" is the only keyword in the question and therefore the highest scoring class is Analyzing, which is given as the prediction result for this question. With this method situations can arise in which a sample has several highest scoring classes with the same score, in the replicated study from Chang et al. [1] this is considered a partial match if the correct class is in the set of highest scoring classes. Unfortunately the lists of common keywords that Chang et al. [1] have used aren't available anymore. Therefore we've replicated the research with 5 different common keyword lists (see section B.4). This left us with a baseline that scored 40% accuracy on the test set. Often this classification method would return more than one predicted class. This classification method scored 53% when we considered the correct class being among the list of predicted classes as being a correct prediction.

The class to which a keyword such as "relate" belongs to can depend on the used list of common keywords. Keywords can even belong to multiple classes in one list. We think that a classifier can perform better if not only keywords are taken into account when classifying but also the context of keywords. The use of N-grams can give more insight to what the context of the question is. Other features, such as sentence lengths, or the presence of numbers, might also have added value. These extra features might help narrow down the actual class.

3.1.3. DATA ACQUISITION

A classifier is useless without data to train on, therefore data acquisition is essential for this project. We looked for labeled datasets and we've found one by Yahya et al. [7]. After careful inspection the dataset turned out to have some badly structured questions and questions that didn't make syntactical sense. For example: "When did you born ?", "How many ways can you a piece of paper?", "Construct simple solutions when problems are complex?". The dataset also seemed like it didn't accurately represent real questions that one would find in university exams. Most questions clearly have one keyword present, often at the very beginning of the question. We had a shortage of data and considering acquiring and labeling data is a time consuming task we decided to correct the spelling and grammar of the questions in this dataset and removed questions that were not representative of real exam questions (such as the examples provided in this subsection). We further expanded the dataset by labeling questions from past university exams to firstly get more data and secondly get a dataset that better represents realistic questions. In total we spent more than 4 full work days per group member to acquire the dataset we currently have. The final dataset consists of 590 prelabeled questions, corrected and curated by us [7], and 919 questions labeled by us.

3.1.4. FEATURES

Whilst labeling we discovered some features that are specific to particular classes in the taxonomy. In subsection 3.1.1 we explained why we think it is necessary to add more features to the vector representation of a question. To measure if a feature is related to a class we chose to use the chi-squared error of the feature per class in the training set. This

Table 3.1: Dataset Samples per class

	Prelabeld [7]	Labeled by us	Total
Remember	104	184	288
Understand	98	248	346
Apply	83	195	278
Analyze	103	120	223
Evaluate	97	118	215
Create	105	54	159
Total	590	919	1509

metric looks at how dependent the occurrence of the feature and the class are. When a feature scores a low chi squared error on all classes then the presence of the feature has little or no added value. We used a 5% chi-squared error as cutoff value for support of the hypothesis that the feature is dependent. The outcome gave us a feeling for which type of features work well and we used this to decide which features might be useful to engineer next. When running an evaluation with our test set we obtain the following scores:

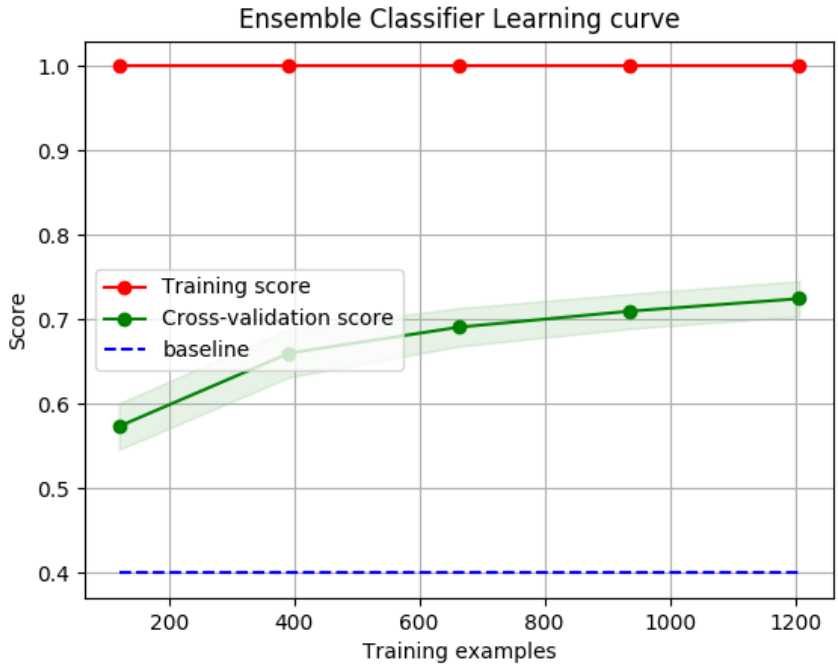
- Using all features: accuracy 0.72, run time 20.54 seconds
- After removing 11 features using a chi-squared error cutoff of 5%: accuracy 0.74, run time 14.16 seconds

Not only do the 11 additional "independent" features lower the cross validation score, they also slow down the training and prediction process of our model.

3.1.5. ACTIVE LEARNING

We plotted a learning curve using 1000 samples of the training data, which indicated that our model suffers from high variance which could be solved with more data. Labeled data however is hard to come by we've noticed. We also noticed that labeling data takes significant effort. Gathering many questions is easier than labeling each one of them so we implemented active learning [5] to help our client efficiently label new questions. The client can enter unlabeled questions into the database. The classifier then tries to predict the classes of these unlabeled questions. The prediction where the predicted class has lowest predicted probability can be labeled by means of active learning to quickly enlarge the labeled dataset in an efficient manner.

Figure 3.1: Ensemble classifier accuracy per number of samples



3.1.6. ONLINE LEARNING

When a teacher labels a question he or she likely wants to see the effects of this extra labeled sample as soon as possible in the results of following classifications. Therefore we've partially implemented the capability for the model to train batches of new samples in an online manner. Retraining the entire model can take significant time if the dataset is large. Because not every machine learning classifier from sklearn implements the function partial _fit, which allows for online learning, we've come up with types of two retraining cycles. In the short cycle all classification algorithms that implement partial _fit are immediately retrained when the desired batch size is reached. In the long cycle all classification algorithms are fully retrained on all available labeled data.

3.1.7. ENSEMBLE CLASSIFICATION

We had to choose a machine learning algorithm to classify our data. We ran a grid search on 7 different classification algorithms on our validation set (20% of the entire dataset) to select the best model. We then ran each classifier on the test set (20% of the entire dataset). The outputted metrics are in the tables below this paragraph. We noticed that some classifiers outperformed the other in a particular class while performing poorly on others and vice versa. To further improve the accuracy of our model we've made use of

an ensemble [6] classifier that uses the prediction of multiple classifier as majority vote to classify. We've chosen classifiers which on their own already perform reasonably. The ensemble classifier (75% accuracy) slightly outperforms the best performing classifier (Passive Aggressive classifier 73% accuracy) on it's own.

3.1.8. PRECISION VS RECALL VS F1 SCORE VS ACCURACY

We looked at 4 different metrics to evaluate our model. Precision, recall and F1 scores are useful stats for having more insight into which classes a model has difficulties predicting. The accuracy is the ratio of the count of correct predictions over the total amount of predictions performed on the test set. We used the accuracy score to compare models' performances, because we want the classifier of our final product to make the most correct predictions given a set of samples. Since the class distribution of our dataset is slightly out of balance (i.e. less "Creating" samples) we also take into account the F1 score of the evaluated models per class, as we would like both a high precision and recall. A high precision and recall per class is desirable as this signifies that the classifier can predict samples of all classes instead of being a lot better at predicting some classes than others.

REFERENCES

[1] Wen-Chih Chang and Ming-Shun Chung. Automatic applying bloom's taxonomy to classify and analysis the cognition level of english question items. In *Pervasive Computing (JCPC), 2009 Joint Conferences on*, pages 727–734. IEEE, 2009.

[2] M Ikonomakis, S Kotsiantis, and V Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

[3] Daniel Jurafsky. Speech and language processing: An introduction to natural language processing. *Computational linguistics, and speech recognition*, 2000.

[4] Anthony Khoo, Yuval Marom, and David Albrecht. Experiments with sentence classification. In Lawrence Cavedon and Ingrid Zukerman, editors, *Proceedings of the 2006 Australasian language technology workshop*, pages 18–25, 2006.

[5] Andrew McCallum, Kamal Nigam, et al. Employing em and pool-based active learning for text classification. In *ICML*, volume 98, pages 359–367, 1998.

[6] K. A. Osadi, M. G. N. A. S. Fernando, and W. V. Welgama. Ensemble classifier based approach for classification of examination questions into bloom's taxonomy cognitive levels. *International Journal of Computer Applications*, 162(4):1–6, Mar 2017. ISSN 0975-8887. doi: 10.5120/ijca2017913328. URL http://www.ijcaonline.org/archives/volume162/number4/27228-2017913328.

[7] Anwar Ali Yahya, Addin Osama, et al. Automatic classification of questions into bloom's cognitive levels using support vector machines. 2011.

4

Classifier Design

4.1. Ensemble

Since choosing a model was not an obvious choice due to several models having similar accuracy, an ensemble classifier was chosen. After evaluation of the classifiers at the end of the project only two classifiers contributed to the ensemble classifier: the passive aggressive and the stochastic gradient descent classifiers. The rest of the models we tried had inferior performance and slowed down the ensemble classifier's prediction and training.

4.2. Support Vector Machine

A stochastic gradient descent classifier (SGD) is a Support Vector Machine [3] (SVM). It is a regularized linear model using the gradient of the loss of each sample to update itself along the way with decreasing strength, so this classifier can be used for online learning. SVMs work well with small datasets such as ours as is discussed by Forman et al. [2].

4.3. Passive Aggressive

The passive aggressive (PA) classifier achieves the highest scores out of all the classifiers we evaluated for our final setup. It has the highest accuracy, recall and f1 score. During the course of the project the PA had similar results to the multinomial Naive Bayes and Stochastic Gradient Descent classifiers, however when our dataset grew larger (i.e. from 600 to 1200 labeled questions) and had more variation, the PA outperformed the other classifiers. The dataset we started with was partially taken from another Bloom's taxonomy classifier project and was highly fabricated with keywords hinting towards a specific Bloom's Taxonomy class present in many samples. When our dataset grew larger with more diverse samples the PA classier and SGD started to stick out from the rest with the PA classifier consistently achieving higher scores.
A PA classifier was not an obvious choice because it is an online classifier that is usually applied to very large datasets [1] that cannot be saved (e.g. twitter feeds). It takes a sam-

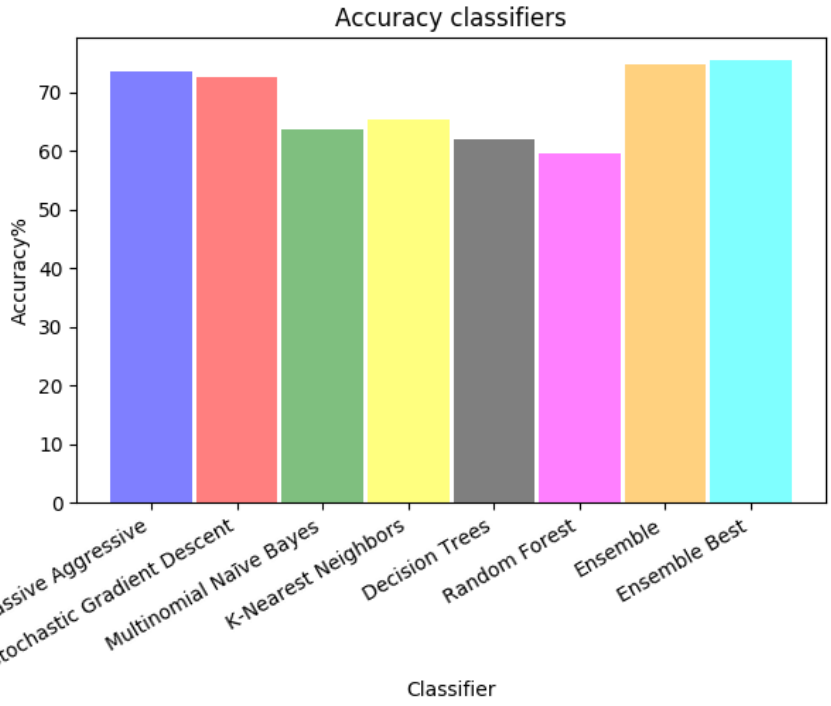


Figure 4.1: Accuracy per classifier
Source: All classifier have been tweaked using hyper parameter tuning

ple, checks the prediction, if correct it does nothing (passive) however if the prediction is incorrect it adjusts the decision boundary function. When a sample is incorrectly classified the PA classifier uses a loss function (without a learning rate scalar as many other classifiers use) and updates the decision boundary so that the incorrectly classified sample would be predicted correctly by a unit margin.
A grid search revealed the aggressiveness parameter C to be optimal for C = 2.5. A large value of C (e.g. C = 100) allows for a fast progress rate, meaning the algorithm adapts well to a small dataset [1]. A small value for C (e.g. C = 0.001) is more appropriate for large datasets (C = 0.001 and C2 = 100 perform similar for a dataset of 3000-4000 samples in the experiment in [1]) but has poor performance for small datasets because each update only changes the online hypothesis by a small amount.
The PA classifier is more robust to noise in samples and labels than linear models [1] (e.g. SVG) according to the experiments conducted in. Since our dataset unavoidably contains incorrectly labeled samples and outliers it is expected that the PA classifier performs well in these conditions. An example of an outlier is: “Provide design guidelines which can be used to guide the development of an architecture containing both pro-

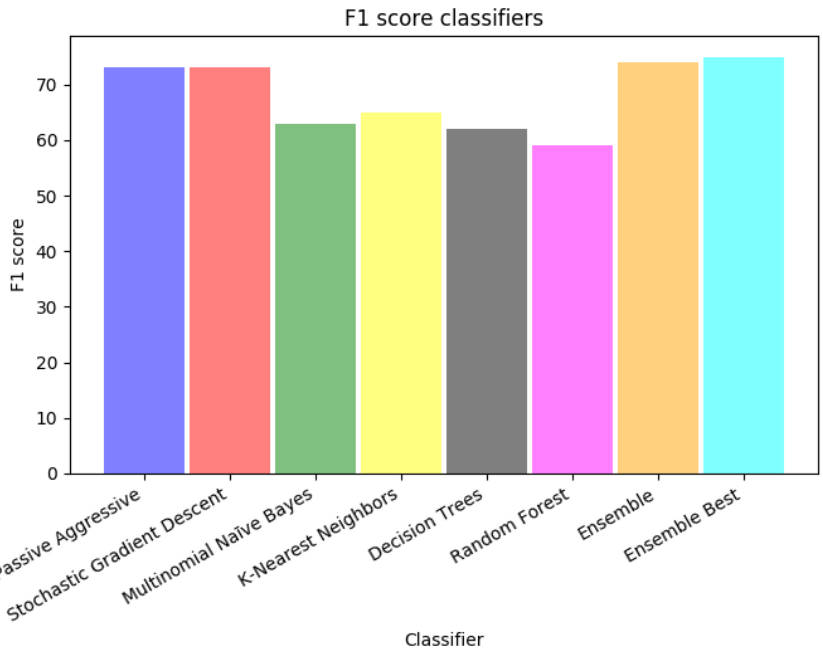


Figure 4.2: F1 score per classifier
Source: All classifier have been tweaked using hyper parameter tuning

cesses and (knowledge) rules. Provide at least 3 guidelines, describe them using the elements of a design guideline and explain their use." One of the reasons why this question is an outlier is because it is of the class Creating but contains keywords of different classes i.e. "explain" and "describe" are keywords of the class Understanding.

REFERENCES

[1] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar): 551–585, 2006.

[2] George Forman and Ira Cohen. Learning from little: Comparison of classifiers given little training. *Knowledge Discovery in Databases: PKDD 2004*, pages 161–172, 2004.

[3] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

5

IMPLEMENTATION

Our software runs on a online server and can be interacted with using our API. We also created a GUI to showcase the systems' capabilities. The GUI is accessed in the browser and consists of two parts: The part that implements the classifier and the part that implements active learning.

5.1. TECHNICAL CHOICES

5.1.1. PYTHON3.5+

When implementing a programming project a language (or set of languages) must be chosen. Considering that the group members were relatively new to machine learning a choice was made to use Python as base language to write the project in. Python is often used to create quick prototypes [6] for scientific projects due to its ease of use and useful libraries (e.g. numpy and matplotlib). Python fits well with our implementation approach allowing us to quickly experiment while learning how to use libraries that are new to us. We made an effort to respect the PIP 8 python naming conventions and to write the code in pythonic style - for example by making use of Python's one-line for loops to build up lists. Our client FeedbackFruits makes use of Python 3.5 and newer. After checking if the necessary libraries (sklearn, numpy, nltk) are compatible with version 3.5 and newer we decided to comply with our client's standard and use Python 3.5 and newer instead of a release of Python 2.

5.1.2. SKLEARN

The sci-kit learn library was used for the machine learning aspects of this project. The library contains each component that we need for the project. The library also has extensive documentation with well written examples. We've inlcuded a list of all sklearn modules we have used in our project (see section B.2).

5.1.3. NLTK

The Natural Language Toolkit was used for preprocessing and part of speech tagging. We made use of the nltk built in text processing functions to convert a sentence to lower case, convert nouns and verbs to base form via stemming and lemmatization, grouping of numbers and variables (i.e. replace them with "NUM" and "VAR" in the question) and remove stop words. At the start of the project we implemented all these functions as part of the preprocessing step. When working on feature engineering we noticed that only the preprocessing step that converts sentences to lower case has a positive impact on the cross-validation accuracy score. Part of speech tagging was needed for some features e.g. “has number any form” performs a word tokenization, then converts the list to part of speech tags and looks if the “NUM” tag is present. We used the “Universal” and “Penn treebank” tag sets and databases. The “Universal” tag set consists of only 12 tags whilst the “Penn treebank” tag set consists of 36 tags which result in more specific tags. We used the “Penn treebank” for most of the situations where part of speech tagging was needed apart from when checking for the presence of any form of number (e.g. 5 or “Five”) we used the “Universal” tag set’s “NUM” tag. To make a feature that removes nouns at the end of noun phrases we followed the method used by Kim SN et al. [5] and used a the nltk chunker parser and the grammar used by Kim SN et al.

5.1.4. POSTGRESQL

In our choice of database we considered both a relational database and a document based database. While document based databases like MongoDB can be easier to set up since they don’t require a structure to be defined beforehand. We decided to go with the relational database PostgreSQL instead. Considering that the software is going to be used in an online learning environment where data will be added continuously, it is safe to assume our data will get very large. Depending on how it will be used, the feedback of teachers on the classification of their questions consists of relational data that can be better represented in a relational database. The use of PostgreSQL over MySQL was decided to accommodate the current data structure of the client.

5.1.5. AMAZON S3

Training the classifier on all the questions in the database can take a long time. We therefore decided to store our trained classifier as a file. This posed the problem of someone being able to access the questions in our (or our client’s) database. To solve this we implemented amazon’s S3 storage service into our software. After training a model our software will upload the classifier via a secure connection to the amazon servers. During runtime the software will use this classifier to predict questions. This method gave us the added benefit of being able to deploy multiple instances of our software without each having to update it’s classifier separately.

5.2. CODE TOOLS

5.2.1. GITHUB

We used Github as version control for our project. We adhered to the git flow branching model as described by Vincent Driessen[7]. For us this meant that each addition to

the software was pushed to a branch in the feature map (branches with a names like `feature/*`). These feature branches were always branched off from the development branch and could only be merged to this branch. The master and development branch were protected and could only be pushed to by pull requests. These pull requests required a successful TravisCI build and at least 1 accepted review by a team member other than the creator of the pull request.

5.2.2. CLOUD PLATFORM

Deployment of our web service is done via Heroku. We chose this cloud platform mainly because of our client's familiarity with it. Heroku also supports postgreSQL databases which made it easy for us to integrate our software into their platform.

Deploying our software on Heroku can be done very quickly and easily by pushing a stable branch to Heroku instead of origin. The configuration file containing the instructions to start the Gunicorn server should be recognized automatically during deployment. The environment variables need to be set to the credentials of the Heroku database and amazon bucket as described in the github readme. If the classifier is not present or needs to be updated the software will do so automatically.

5.3. CLASSIFICATION GRAPHICAL USER INTERFACE (GUI)

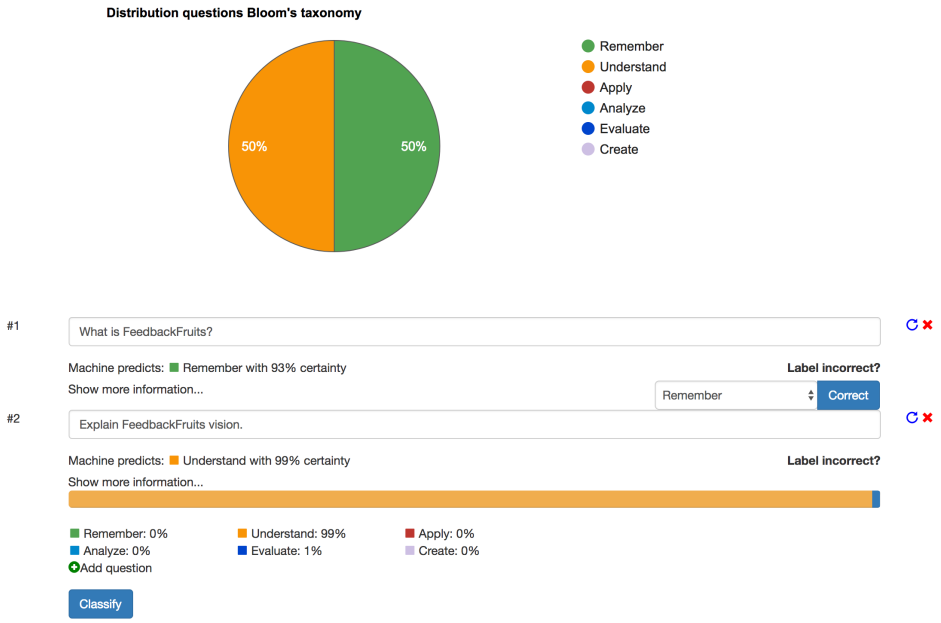


Figure 5.1: Screenshot GUI

In the part of the GUI that implements the classifier we have given the user the ability

to add the question set they want to classify in the input fields. When the classify button is clicked an API call is made to the classifier in the back-end. The classifier will then classify each question according to Bloom's revised taxonomy and return the probabilities it assigns to each class. Under the inputted questions a prediction will appear with a certainty level. The user can click "show more information" to see the percentages the classifier has predicted in the other classes of the taxonomy. At the top of the screen a pie-chart representing the distribution of classes will appear. Each class is color coded so the user can easily identify which predicted classes are present in the pie-chart and in the bar that shows up when a user clicks "show more information".

The classifier is not perfect, and sometimes might give the wrong classification. In this case the user can click "Label incorrect?" to give the classifier feedback. This feedback will be incorporated in the next time the model is trained, or if the batch size for online learning is reached. The batch size can be set by the client, we've set the batch size at 5 relabeled samples before the classifier refits these samples.

When there isn't a clear difference in the percentages there will be an indication next to the question that it is deemed ambiguous. The user can then edit the question in place and then predict it again by clicking the blue repeat button next to the question. The class with the highest certainty will be represented in the pie-chart, whereas the class with the second highest certainty will not.

To help our client with labeling data in an effective manner we've development a GUI

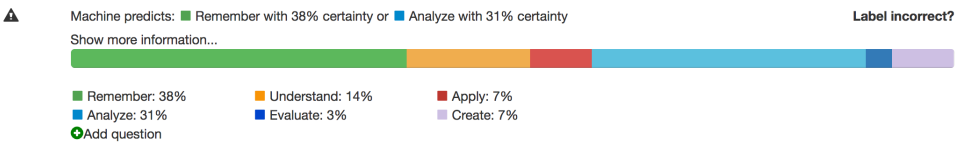


Figure 5.2: Example GUI ambiguous question

that implements active learning. The user gets to see an unlabeled question and the accompanied probabilities the current classifier assigns to each class. Based on the question and shown probabilities the user can then label the question by selecting a label and click "correct" or "skip" to skip the question if the annotator does not know the label at the time.

5.3.1. ETHICAL IMPLICATIONS

The target audience for this project is teachers that are looking to align their courses according to Bloom's revised taxonomy. The classifier we have developed isn't perfect. It is possible that the classifier predicts an incorrect class for a question. Teachers may use this incorrect information to align their course. This could result in a misaligned course and could potentially decrease learning in students. We have added a section in the GUI stating that the prediction might be incorrect to ensure that the teachers using our classifier understand that the resulting prediction may be incorrect and that they

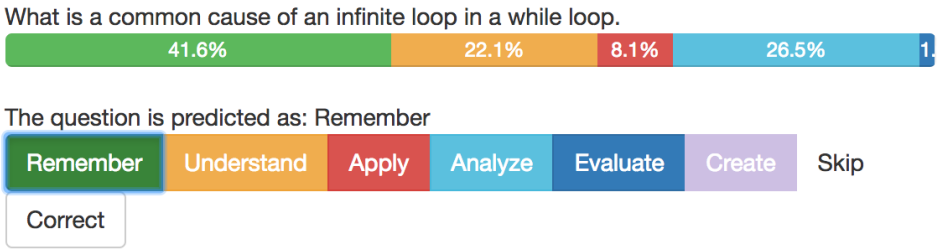


Figure 5.3: Active Learning GUI

should use this information at their own discretion.

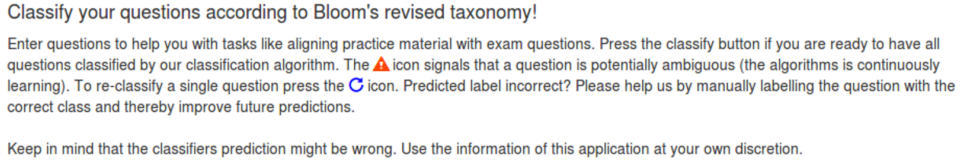


Figure 5.4: Description GUI

5.4. CODE STRUCTURE

The source code can be divided in two main parts: the front-end and the back-end. The code itself is also divided in 2 packages (`app` and `classifier` for the front-end and back-end respectively)

The project root contains project specific configurations (such as deployment info for Heroku, requirements for a pip dependency install, travis , codecov and nosetests configurations).

5.4.1. FRONT-END

The front-end of our software consists of the API and a simple GUI showcasing our software's abilities. Both are created using Flask and as such is structured as suggested in: Flask Web Development by Miguel Grinberg. [4].

Though our software is supposed to be used by using the API, the structure suggested by Grindberg [4] allows for easy extending of the front-end. More information on this can be found in the Github readme of our project.

5.4.2. CLASSIFIER

The classifier package contains the ensemble classifier as well as the features used and all functions used by the API to train and use the classifier. The package also contains

the online and active learning components of our software. The package is divided into 4 smaller packages: `base`, `features`, `utils` and `web`. The rest of this section will explain in short which parts of our software are contained in these packages.

BASE

The base packages contains the classifier definition and it's most important functions. The online and active learning components are also located in this package. Extending the core functionality of our software requires working with this package.

FEATURES

The features packages contains all the features used by our models, as well as all the text processing steps needed to use them such as stemming and lemmatization.

UTILS

The utils package contains code that is used in multiple places in the project, it also defines some configuration settings for sklearn and the database interface.

WEB

The web package contains the database interface that allows our software to python functions to do things like retrieving questions, adding them to the database or labeling them. Running the software locally also requires the database and amazon s3 credentials to be placed in this package.

5.5. TESTING

Testing is essential to having a maintainable project. During the course of this project we adopted an experimental hands on approach to writing the code. This means that when implementing different aspects of the project that we had no prior experience with, we had to experiment and try out various implementations before choosing a definitive one. Since we did not always know how to implement new features we did not opt for a test-driven-development approach but only wrote tests once we were more or less satisfied with our implementation. Furthermore our project makes use of the sklearn machine learning library. In sklearn, certain objects such as classifiers have learned parameters which can make it challenging to write meaningful tests.

5.5.1. TOOLS

We used the 'nosetests' plugin to run all our test code both locally and on TravisCI. During the course of our project we also started using "code climate" (an automated code review tool) and in the final weeks 'Codecov' which displays the difference in test coverage every commit (to see if our test coverage improves or worsens per pull request). These tools don't only help with getting feedback on the status of our testing suite but also make improving the code and writing tests a more rewarding task.

5.5.2. TEST TYPES

To correctly and more easily maintain a large software project, different kind of tests can be useful. In our software we make a distinction between Unit tests, Integration tests and functional tests.

UNIT TESTING

The bulk of our tests are unit tests, i.e. small pieces of code testing individual functions. Tests for the features of our classifier as well as some of the preprocessing steps are mostly unit tests.

In most of our other functions mocks are used to keep the unit tests and the part of the code that they cover as small and separated as possible.

INTEGRATION TESTING

Not all tests can use mocks though. Our software uses several different components (postgreSQL database, sklearn components, amazon S3 buckets, nltk libraries, etc.). To test the integration between our software and those components we use integration tests.

The database-interface tests for instance uses a schema created from the actual database to create a test instance of a database at the start of the test suite. This way database manipulation can be tested without needing to change the production database.

FUNCTIONAL TESTING

Lastly we have functional tests that test complete parts of our software. For the API we use a Flask test client to test the functions of our API. The classifier is tested by training a model on our training data and evaluating prediction results, making sure the accuracy is above a threshold set at 40% (threshold chosen at accuracy of baseline).

CONTINUOUS INTEGRATION

During the project we made use of a continuous integration service called TravisCI to make sure that new features or code changes did not break anything. TravisCI (for private repositories) is offered for free with the github student pack for which we subscribed.

We set up our continuous integration service to automatically submit our code to an automated code review service called codeclimate. This helped us prepare for the SIG evaluation as well as giving us a overview of how our code coverage changed with each pull request.

Our TravisCI configuration uses a clean version of the Ubuntu 12.04.5 LTS precise operating system. All required software and packages are downloaded and installed

for each release. This environment is similar to cloud platform environments such as Heroku. This helped us develop with deployment to a cloud platform in mind, since a successful Travis build was required for each pull request.

5.5.3. SIG

Halfway the project the code was submitted to SIG for an automated code review. At that time the test coverage was slightly above 80%. SIG submitted positive feedback on the test code and advised us to continue writing tests for code written after the 1st SIG deadline. The majority of the code was written before the SIG deadline, it was not challenging to keep a relatively high test coverage.

5.6. DATABASE

Our software needs to be able train on a large dataset of questions. The dataset is stored in an online database for ease of access and this is how our client wishes to access the dataset. The database contains the questions as well as the labels that our classifier trains on. It also contains the names of all users that have contributed as well as the sources, e.g. “TU Delft - Multimedia Analysis exam 2015” of their questions in the case that they didn’t come up with it themselves. Users should be able to add questions to the database and be able to label the questions that are in the database.

A visualization of our relational postgreSQL database schema is given in Figure 5.5 and the rest of this section will explain the structure as well as the choices made in defining the database structure in more detail.

5.6.1. ONLINE DEPLOYMENT

At the start of the project the database was hosted locally on a postgreSQL server to allow for fast data transfers. When we started deploying the application on Heroku (see subsection 5.2.2) however, we decided to use the postgreSQL database service they provide. This way we did not have to keep a server online ourselves, but since a Heroku postgreSQL database can also be accessed from applications not running on their servers we could still access it in the same way we did before.

Unfortunately the free plan on Heroku only allows a user to keep two backups of their database. We felt that since our database changed a lot during the project, it was necessary to have a more complete history of our database. To achieve this we set up a raspberry pi to create and store backups of our database at regular intervals.

5.6.2. ENTRIES

When a user corrects a predicted label or the client enters a label for a question is it saved in the database as an entry. It is possible that some users disagree about the classification of a particular question. In such a case it is preferable to keep both user’s input but to decide on one classification for our classifier to use.

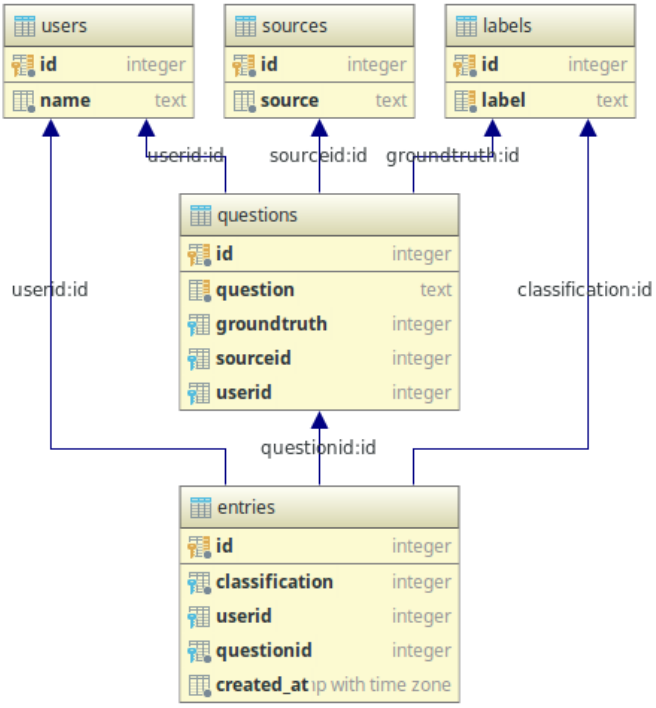


Figure 5.5: The database structure of our software.

For this reason we have decided to define a table called entries, where each entry represents a label given by a user to a question. Instead defining the label in the questions table each entry has a foreign key that references the id of the question it classifies.

Since each entry is made by a user the entries table also contains a foreign key referencing the users table and a time-stamp to be able to select entries from specific dates.

5.6.3. QUESTIONS

The questions table stores questions as text with foreign keys referencing a user and source for each question. This makes it easy to see who entered the question into the database and if it came from a respectable source. When making the initial dataset for this project these keys were especially useful to see that there was no overlap in our sources.

Since our database allows for multiple users to label a single question, but our classi-

fication algorithm requires a singe value, we defined a single column (called 'groundtruth') to hold the final label that should be used by the classifier. Before training the classifier this column is generated from the entries in the following way:

- For each question all entries are counted, a entry for a question with a classification of Understand counts as a vote for Understand on that question.
- The label with the most votes is considered the 'groundtruth' for that question
- In the event of a tie between 2 or more labels, the label with the newest entry is considered the 'groundtruth'

5.6.4. LABELS

The labels table contains each class in the blooms taxonomy paired with an id. This is so that the entries and question tables can use a foreign key as reference. This prevents errors that can be caused by mistyping a classification when editing the database manually. This also makes it possible to edit the Bloom's taxonomy classes without having to edit each row in the entries and questions tables.

5.7. WEB SERVER

A goal of our software is to run on any server or cloud platform (supporting python) out of the box, allowing developers to use our classifier via an API. Since we use sklearn in python for most of the machine learning operations, we decided to also write the Webserver and API code in python.

The Flask web framework for python allowed us to develop a web application relatively quickly, with its integrated development server we did not have even have to integrate a production HTTP server until later into the project.

To deploy our software we chose to use the web server gateway interface (WSGI) Gunicorn. This choice was made because of it's compatibility and ease of use with the Flask framework. The Flask built in server was easy to set up during development, but can only handle one request at a time, greatly increasing the loading times when multiple users used our software. Gunicorn performed better, reducing loading times even on single requests.

5.8. API

To use our classifier developers are able to use functions in our API. These functions allow a user to add questions to the database, label or re-label them, re-train the entire classifier on the new data and of course predict questions.

After discussing the required functionality with the client we decided to follow the JSON-API style[3]. Since python dictionary objects a easily parsed into or from JSON we decided on a JSON based API. JSON-API has guidelines covering creating and updating resources whereas other API guidelines such as JSEND[2] or HAL[1] focus more on defining guidelines for responses.

More information on the usage of our API can be found in the documentation.

5.9. SIG

On the first of June we submitted a preliminary version of the project to the Software Improvement Group (SIG). Our code on the first of June scored 3 stars on SIG's maintainability model. Main points for improvement where Unit Size and code duplication. Which means that the size of some of our methods were above average, and some methods had code in them that could be found in other methods also.

5.9.1. HANDLING FEEDBACK SIG

We used the static code analysis tool CodeClimate to get a more detailed overview of improvements that could be made to our code. This showed that indeed the Unit size was way to long for some code in our templates map. These are standard bootstrap templates however so we did not write those ourselves. Including this code was a mistake on our part. This code will not be included in the final version since we rewrote the GUI to more clearly show the features of our software.

Code duplication was still an issue however. CodeClimate helped us in this regard by notifying when duplicate code was found after each pull request.

REFERENCES

[1] Hal - hypertext application language. URL http://stateless.co/hal_specification.html.

[2] Jsend. URL http://labs.omniti.com/labs/jsend.

[3] Json api a specification for building apis in json. URL http://jsonapi.org/.

[4] Miguel Grinberg. *Flask Web Development*. OReilly, 2014.

[5] Su Nam Kim, Timothy Baldwin, and Min-Yen Kan. Evaluating n-gram based evaluation metrics for automatic keyphrase extraction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 572–580. Association for Computational Linguistics, 2010.

[6] K Jarrod Millman and Michael Aivazis. Python for scientists and engineers. *Computing in Science & Engineering*, 13(2):9–12, 2011.

[7] Vincent Vincent Driessen. A successful git branching model. URL http://nvie.com/posts/a-successful-git-branching-model/.

6

Reflection

In this section we will reflect on how the project went and explain why some software solutions didn't end up being part of the project.

The original proposed project was to generate feedback for assignments much like a peer review system. Our client gave us access to a dataset of around 20,000 samples however there was no consistency in the data. Samples were not written in the same language, feedback was not always serious or correct, and for the feedback to be interpreted one would often need more information than available in the data sample. Besides this, our supervisors warned us that this was a complicated problem to solve, so we decided to move on to finding another problem to solve.

The second proposed project was to generate questions each level of Bloom's taxonomy, named "Automatically generating practice exam questions". In the research phase we realized that this project was infeasible, and that it very much is an open question in the field of AI. We attempted to generate questions by fitting templates to Wikipedia articles. This resulted mainly in questions in the two lower levels of Bloom's revised taxonomy. Questions higher up in the taxonomy tend to revolve around abstract concepts. We tried to generate questions higher up in the taxonomy by using word embeddings to see how close a word vector was to for example the word vector technology. We could also use this distance to create analyze questions for example compare x an y, where x and y are two vectors that are close to each other. The distance between word vectors however, isn't a guarantee that words are related. The questions make lexical sense, however they don't always make semantic sense. The amount of bad questions would be high. To solve this we would have to build a classifier that discerns between good and bad questions, for which we would need an AI that understands semantics. Getting a machine to understand semantics is considered an AI-hard problem.

We then tried to come up with a different way to get a hold of questions in an automated fashion. Scraping questions from the web came to mind. Building a scraper turned out

to be fairly easy. Finding questions on the other hand turned out to be hard. First off the questions that one finds on the open web are very different than the ones you find in education. In eduction teachers can ask imperative questions of various levels in the taxonomy. On the web people mostly ask questions in the lower levels of the taxonomy. Many questions posed on the open web also aren't questions you would see in education. One option to combat this was to scrape educational websites. This however brought some copyright concerns with it.

6.1. PROCESS

Working in a team of three members requires coordination. We opted for a fast prototype approach since we were working in a field in which we had little prior experience. Instead of researching all possible solutions and picking the one that we thought would work best, since we did not know before hand which solution would work best we decided to experiment and try several solutions. Therefore we made more features and evaluated more models than what we kept in our final ensemble classifier. We wanted to prototype fast so we could showcase our results to our client and get feedback about what exactly they expected from our product. Overall the client was satisfied and sometimes even excited when we presented demos of our product. FeedbackFruits has biweekly presentations on Tuesday nights on which we presented prototypes at two different stages to get feedback. During one of these sessions we got feedback about how to handle questions of which the class is not obvious (see subsection 2.1.2).

FeedbackFruits welcomed us to work at their office. We worked at their office approximately from 9-17 or 10-18 during weekdays. Working in the client's office had positive impact on our progress. When encountering a problem or when uncertain about an implementation choice we scheduled meetings with FeedbackFruits's employees. We held a weekly meeting with our supervisor Pim de Haan who gave us useful insights on all areas from machine learning to database design choices.
newline Since we had to learn more about machine learning and had to build our own dataset we decided to also work at the office during public holidays and during most of the weekends. Every day we would make a list of task priorities that we needed to finish for our next prototype. We would often work on one feature with more than one person so that we could discuss how to implement it. As we would sit next to each other and see each other's screens we made a habit of quickly helping other group members when they were stuck or were implementing data structures that another group member was familiar with. Even though we worked together we each put more time into different aspects of the product: Dennis specialized in the webserver and database implementation and code tools, Joe specialized in making the ensemble classifier, online learning and active learning features and Olivier specialized in the feature engineering and model evaluation.

7

Future Work

In this section we will first discuss the limitations of our project. We will then recommend way to solve or mitigate these limitations in the future.

7.1. Discussion

The classifier that we have developed for this project doesn't always give the right prediction. In the problem definition we have limited the scope to correctly spelled standalone, American English questions. In this section we will discuss the limitations of our classifier pertaining to the scope of our project and a few examples that are within the scope but still fail to give the correct prediction.

Misspelled words can be detrimental to the accuracy of the prediction. For example the correctly spelled question: 'Label the parts in the diagram' is predicted having the label Remember. However, the misspelled question: 'Labl the parts in the diagram' is predicted having the label Understand. As discussed in the section on creating a baseline, keywords play an important role in predicting the correct class. Our model does not correct misspelled words and therefore the classifier lacks information on the presence of important keywords when making a prediction. Correctly spelled British English questions can also be wrongly classified due to the same reason. For example the question 'Analyse the test results.' is incorrectly classified as a Creating question.

Due to a shortage of data our model does not generalize well. An example of this is the question "Subdivide the problem into its core components." which is classified as an Apply question while it should have been classified as an Analyze question. The dataset of labeled questions does not contain a question that contains the word subdivide. That's why the classifier wrongly predicts this particular question.

7.2. RECOMMENDATIONS

The gap of the learning curve in figure 3.1 indicates that it is useful to expand the dataset. We haven't come across all common keywords from the lists in B.4 Finding more samples would help to generalize our model so that cases where question containing the keyword subdivide are predicted correctly. We also recommend phasing out the dataset created by Yahya et al. [1] once the entire dataset has been expanded sufficiently to get a classifier that is trained on more realistic data.

To solve the problem of incorrectly predicting the class of misspelled and question using British English spelling we could preprocess the data such that each question is correctly spelled in either British or American English by means of a automatic spell checker. Spell checkers could however incorrectly correct words, e.g. 'ame two presidents' could become 'fame two presidents' instead of 'name two presidents'. We do think that British English spelled words could easily be converted into American English.

The preprocessing and features used are the same for each classifier. Some classifiers may perform better with different features and preprocessing steps and parameters. The search space for this optimization is very large. We propose using genetic programming to converge to an optimum.

REFERENCES

[1] Anwar Ali Yahya, Addin Osama, et al. Automatic classification of questions into bloom's cognitive levels using support vector machines. 2011.

8

Conclusion

8.1. Conclusion

The client of this bachelor project FeedbackFruits requested a classifier that classifies questions according to Bloom's taxonomy. We built a classifier that predicts the correct class out of the six possible taxonomy classes 75% of the time in our test set and outperforms the baseline we set. We have also provided FeedbackFruits with a labeled dataset with 1509 questions labeled according to Bloom's revised taxonomy. We have provided an interface to efficiently expand the dataset of questions by means of active learning. To showcase the capabilities we have created a GUI. Teachers are able to give feedback on the correct label of a presented question, this feedback is then inserted into the database and will be taken into account by our classifier when the model is retrained. The product was well received by our client.

Appendix A shows our project requirements in a MoSCoW model. The must-haves and should-haves all requirements have been satisfied. The could have question generation has been abandoned in the research phase (as described in chapter 6). We also did not build a GUI in the style of a FeedbackFruits webservice, but did create an API to their specifications. This allows our client to implement the classifier in the way they see most fit.

REFERENCES

[1] Hal - hypertext application language. URL http://stateless.co/hal_specification.html.

[2] Jsend. URL http://labs.omniti.com/labs/jsend.

[3] Json api a specification for building apis in json. URL http://jsonapi.org/.

[4] Nancy E Adams. Bloom's taxonomy of cognitive learning objectives. *Journal of the Medical Library Association : JMLA*, 103(3):152–153, jul 2015. doi: 10.3163/1536-5050.103.3.010. URL https://doi.org/10.3163/1536-5050.103.3.010.

[5] Lorin W Anderson. An empirical investigation of individual differences in time to learn. *Journal of educational psychology*, 68(2):226, 1976.

[6] Lorin W Anderson, David R Krathwohl, P Airasian, K Cruikshank, R Mayer, P Pintrich, J Raths, and M Wittrock. A taxonomy for learning, teaching and assessing: A revision of bloom's taxonomy. *New York. Longman Publishing. Artz, AF, & Armour-Thomas, E.(1992). Development of a cognitive-metacognitive framework for protocol analysis of mathematical problem solving in small groups. Cognition and Instruction*, 9(2):137–175, 2001.

[7] John Biggs. Enhancing teaching through constructive alignment. *Higher education*, 32(3):347–364, 1996.

[8] Benjamin S Bloom et al. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay*, pages 20–24, 1956.

[9] Phyllis Blumberg. Maximizing learning through course alignment and experience with different types of knowledge. *Innovative Higher Education*, 34(2):93–103, 2009. ISSN 1573-1758. doi: 10.1007/s10755-009-9095-2. URL http://dx.doi.org/10.1007/s10755-009-9095-2.

[10] Phyllis Blumberg. Maximizing learning through course alignment and experience with different types of knowledge. *Innovative Higher Education*, 34(2):93–103, 2009.

[11] Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. Automatic question generation for vocabulary assessment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 819–826, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220678. URL http://dx.doi.org/10.3115/1220575.1220678.

[12] Wen-Chih Chang and Ming-Shun Chung. Automatic applying bloom's taxonomy to classify and analysis the cognition level of english question items. In *Pervasive Computing (JCPC), 2009 Joint Conferences on*, pages 727–734. IEEE, 2009.

[13] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7 (Mar):551–585, 2006.

[14] George Forman and Ira Cohen. Learning from little: Comparison of classifiers given little training. *Knowledge Discovery in Databases: PKDD 2004*, pages 161–172, 2004.

[15] Meredith D. Gall. The use of questions in teaching. *Review of Educational Research*, 40(5):707–721, 1970. ISSN 00346543, 19351046. URL http://www.jstor.org/stable/1169463.

[16] Miguel Grinberg. *Flask Web Development*. OReilly, 2014.

[17] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[18] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: The state of the art. *Comput. Linguist.*, 24(1):2–40, March 1998. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972719.972721.

[19] M Ikonomakis, S Kotsiantis, and V Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

[20] Daniel Jurafsky. Speech and language processing: An introduction to natural language processing. *Computational linguistics, and speech recognition*, 2000.

[21] S Kannan and Vairaprakash Gurusamy. Preprocessing techniques for text mining. 2014.

[22] Anthony Khoo, Yuval Marom, and David Albrecht. Experiments with sentence classification. In Lawrence Cavedon and Ingrid Zukerman, editors, *Proceedings of the 2006 Australasian language technology workshop*, pages 18–25, 2006.

[23] Su Nam Kim, Timothy Baldwin, and Min-Yen Kan. Evaluating n-gram based evaluation metrics for automatic keyphrase extraction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 572–580. Association for Computational Linguistics, 2010.

[24] Ming Liu, Rafael A Calvo, Anindito Aditomo, and Luiz Augusto Pizzato. Using wikipedia and conceptual graph structures to generate questions for academic writing support. *IEEE Transactions on Learning Technologies*, 5(3):251–263, 2012.

[25] Christopher D Manning, Hinrich Schütze, et al. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[26] Karen Mazidi and Paul Tarau. Automatic question generation: From nlu to nlg. In *Proceedings of the 13th International Conference on Intelligent Tutoring Systems - Volume 9684*, ITS 2016, pages 23–33, New York, NY, USA, 2016. Springer-Verlag New York, Inc. ISBN 978-3-319-39582-1. doi: 10.1007/978-3-319-39583-8_3. URL https://doi.org/10.1007/978-3-319-39583-8_3.

[27] Andrew McCallum, Kamal Nigam, et al. Employing em and pool-based active learning for text classification. In *ICML*, volume 98, pages 359–367, 1998.

[28] K Jarrod Millman and Michael Aivazis. Python for scientists and engineers. *Computing in Science & Engineering*, 13(2):9–12, 2011.

[29] Assel Omarbekova, Altynbek Sharipbay, and Alibek Barlybaev. Generation of test questions from rdf files using python and sparql. In *Journal of Physics: Conference Series*, volume 806, page 012009. IOP Publishing, 2017.

[30] K. A. Osadi, M. G. N. A. S. Fernando, and W. V. Welgama. Ensemble classifier based approach for classification of examination questions into bloom's taxonomy cognitive levels. *International Journal of Computer Applications*, 162(4):1–6, Mar 2017. ISSN 0975-8887. doi: 10.5120/ijca2017913328. URL http://www.ijcaonline.org/archives/volume162/number4/27228-2017913328.

[31] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[32] Vincent Vincent Driessen. A successful git branching model. URL http://nvie.com/posts/a-successful-git-branching-model/.

[33] Anwar Ali Yahya, Addin Osama, et al. Automatic classification of questions into bloom's cognitive levels using support vector machines. 2011.

A

REQUIREMENTS

A.1. MoSCoW model

Must have:

- A dataset of questions (pre-existing dataset at minimum)
- A classifier that classifies the questions in the dataset according to Bloom's taxonomy.
- the classifier will also give the probabilities of each class
- Active learning component that allows both us and the client to label the most important parts of our/their dataset.

Should have:

- A GUI to showcase the classifier:
  - with an input field to enter the query question and an output section to display the result classification according to Bloom's taxonomy along with confidence levels for each classification in the taxonomy
  - a part that showcases the active learning component
- Online learning for user feedback to be taken into account in our model
- Our software running completely independently on a cloud platform as a webservice
- Our client should be able to use our software via JSON API requests to our webservice
  - Our classifier should be able to predict questions via an API request
  - Questions can be added (and/or labeled) to the database via API request

  - Our software should be able to retrain the classifier via API request without requiring re-deployment

Could have:

- A question generator that augments the training data to more closely resemble the expected use case of our client.
- A production quality GUI in the same style as other Feedbackfruits webservices

Won't have:

- The option to select a language other than English
- Similar questions recommendation
- Search engine to retrieve questions from the database
- Question conversion from one class of Bloom's taxonomy to another class.

B

Additional Resources

B.1. Evaluation results per classifier

All classifiers have tuned parameters acquired via an exhaustive grid search. Following are the precision, recall and f1 score per classifier using stratified splits with a test size of 20%.

Table B.1: Passive Aggressive

	precision	recall	f1-score	support
Remember	0.65	0.88	0.74	58
Understand	0.75	0.67	0.71	69
Apply	0.76	0.8	0.78	56
Analyze	0.79	0.61	0.69	44
Evaluate	0.75	0.56	0.64	43
Create	0.78	0.91	0.84	32
Avg / total	0.74	0.74	0.73	302

Table B.2: Stochastic Gradient Descent

	precision	recall	f1-score	support
Remember	0.61	0.78	0.68	58
Understand	0.71	0.65	0.68	69
Apply	0.69	0.73	0.71	56
Analyze	0.64	0.64	0.64	44
Evaluate	0.69	0.56	0.62	43
Create	0.89	0.75	0.81	32
Avg / total	0.69	0.69	0.69	302

Table B.3: Multinomial Naïve Bayes

	precision	recall	f1-score	support
Remember	0.67	0.76	0.71	58
Understand	0.60	0.77	0.67	69
Apply	0.70	0.80	0.75	56
Analyze	1.00	0.43	0.60	44
Evaluate	0.79	0.35	0.48	43
Create	1.00	0.38	0.55	32
Avg / total	0.73	0.64	0.63	302

Table B.4: K-nearest Neighbors

	precision	recall	f1-score	support
Remember	0.73	0.78	0.77	58
Understand	0.75	0.62	0.68	69
Apply	0.52	0.70	0.60	56
Analyze	0.58	0.66	0.62	44
Evaluate	0.76	0.44	0.56	43
Create	0.61	0.69	0.65	32
Avg / total	0.67	0.65	0.65	302

Table B.5: Decision Tree

	precision	recall	f1-score	support
Remember	0.63	0.69	0.66	58
Understand	0.75	0.65	0.70	69
Apply	0.51	0.52	0.51	56
Analyze	0.64	0.61	0.63	44
Evaluate	0.62	0.60	0.61	43
Create	0.53	0.62	0.57	32
Avg / total	0.63	0.62	0.62	302

Table B.6: Random Forest

	precision	recall	f1-score	support
Remember	0.58	0.88	0.70	58
Understand	0.75	0.52	0.62	69
Apply	0.54	0.59	0.56	56
Analyze	0.58	0.57	0.57	44
Evaluate	0.46	0.40	0.42	43
Create	0.72	0.56	0.63	32
Avg / total	0.61	0.60	0.59	302

Table B.7: Ensemble

	precision	recall	f1-score	support
Remember	0.72	0.88	0.79	58
Understand	0.72	0.71	0.72	69
Apply	0.76	0.80	0.78	56
Analyze	0.76	0.64	0.69	44
Evaluate	0.83	0.56	0.67	43
Create	0.76	0.91	0.83	32
Avg /total	0.75	0.75	0.74	302

Table B.8: Ensemble best

	precision	recall	f1-score	support
Remember	0.68	0.86	0.76	58
Understand	0.75	0.68	0.71	69
Apply	0.79	0.82	0.81	56
Analyze	0.78	0.66	0.72	44
Evaluate	0.77	0.63	0.69	43
Create	0.83	0.91	0.87	32
Avg / total	0.76	0.75	0.75	302

B.2. SKLEARN MODULES USED

- PassiveAggressiveClassifier
- SGDClassifier
- GridSearchCV
- TfidfTransformer
- HashingVectorizer
- Metrics
- LearningCurve
- ShuffleSplit
- BaseEstimator
- TransformerMixin

B.3. FEATURE LIST

??

- Binary features:
  - contains question mark

  - starts with “wh” : looks at first word of input
  - sentence starts with “wh” : if the question is composed of several sentences it looks at the first word of each sentence
  - question length is short : checks if the number of characters in the input is less than the value of the first quartile of the database question length distribution
  - question length is long : checks if the number of characters in the input is more than the value of the third quartile of the database question length distribution
  - question has clause : checks if one or more sentences without a question mark precede the last sentence of the input
  - has “could” “should” “would” : checks for the presence of one or more of these specific keywords
  - has remembering keyword
  - has understanding keyword
  - has applying keyword
  - has analyzing keyword
  - has evaluating keyword
  - has number : [1-9]
  - has number any form : also takes text form numbers into account e.g. “five”
  - has operator
  - has operator after number : e.g. “6+”
  - has operator before number : e.g. “* 9”
- input followed by space separated list of part of speech tags
- term frequency – inverse document frequency
- remove all nouns at the end of noun phrases

The following features are not used in our final model. List of features with low chi-squared scores:

- contains quote pair
- contains semicolon
- contains colon
- question length is between the 1st and 3rd quartiles of the database question length distribution
- question word count

- number of sentences
- question has past tense verb
- question has past participle
- question has verb repetition : verbs are put in base form before checking occurrences
- has operator surrounded by numbers

B.4. SOURCES LISTS OF COMMON KEYWORDS

Azusa Pacific University
http://www.apu.edu/live_data/files/333/blooms_taxonomy_action_verbs.pdf
Cornell University
https://www.cte.cornell.edu/documents/Assessment%20-%20Blooms%20Taxonomy%20Action%20Verbs.pdf
California State University
http://www.fresnostate.edu/academics/oie/documents/assesments/Blooms%20Level.pdf
Missouri State University
https://www.missouristate.edu/assets/fctl/Blooms_Taxonomy_Action_Verbs.pdf
Madison Area Technical College
https://adp.uni.edu/documents/bloomverbscognitiveaffectivepsychomotor.pdf

General Information:
Title of the Project: Question classification according to Bloom's revised taxonomy
Name of the client organization: FeedbackFruits
Date of the final presentation: The 6th of July, 2017

Description: FeedbackFruits is an online platform that helps teachers organize their courses and give them insights on their students' performance. A big problem in education is the alignment of course material and course assessment. To align a courses' components teachers can classify their course material and course assessment according to Bloom's revised taxonomy, a taxonomy that categorizes questions into six cognitive levels of difficulty. In this project we have built a classifier that classifies questions according to this taxonomy to help teachers save time and better align their courses.
The challenge of this project was building a classifier that performs well with a limited dataset and help our client with expanding this limited dataset.
We researched what the best preprocessing steps, features, machine learning algorithms and parameters were for a good classifier. We researched active learning methods to expand limited datasets in a efficient way. We also researched how to design a database and API to use the classifier
We set weekly goals and had daily sprint meetings. Each member of the team had his own area of expertise which he worked on with occasional overlap.
We developed a classifier that had an accuracy of 75% on our test set. We also built a GUI that teachers can use to classify questions and get an overview of the classes present in their course.
FeedbackFruits was very excited about our classifier and will use the API in their existing platform.

Team Roles Name: Joe Harrison
Interests: Artificial Intelligence, Augmented Reality, Computer Vision
Contribution and role: development classifier, front-end design and evaluation
Name: Olivier Dikken
Interests: Computer Graphics, Game Development, Artificial Intelligence
Contribution and role: scrum master, feature engineering, evaluation
Name: Dennis van Peer
Interests: Embedded Systems, Computer Vision
Contribution and role: back-end design, database design, API design
All team members contributed to preparing the report and the final project presentation.

The final report for this project can be found at: `http://repository.tudelft.nl`

C

Research Report

C.1. Introduction

The importance of providing good practice questions is widely acknowledged by education professionals [7]. Practice questions need to reflect the level of understanding required from the student. It is a common practice in education to test different facets of a student's knowledge and understanding. Providing teachers with an overview of the type of questions they pose can help with matching the assessment of their courses' learning objectives with the learning objectives [3].
Bloom's revised taxonomy consists of 6 levels of comprehension ranging from *remember* to *create*. Each category of questions requires a deeper level of understanding to answer correctly [3]. Our project aims to provide a software solution that can classify questions according to this taxonomy
While we will provide a GUI with our project showcasing the functionality and integration of our classifier into online education systems (like FeedbackFruits) is where our project will have the most impact. These systems can implement our algorithm in their existing solutions in such a way that practice exams and assignments can be evaluated in the background to show statistics. Our project will contain an API that these companies can use to extend the functionality of their platform.

C.2. Problem Definition

A big problem in education is that the practice questions provided to students do not always align with the learning goals of the course. An exam for a course needs to be representative of the learning goals [4]. Our software would help the teacher in the following ways:

- Having an algorithm look at the complexity of the questions a teacher has created allows him/her to have a guideline on the cognitive level required by the student to answer the questions.
- Analyzing the complexity of all questions (for example of multiple versions of a

multiple choice exam) can be a time consuming task. A teacher may consider it unnecessary to assess the level of every question appearing in his course material. Our classifier could be used as part of a program that automates this process gaining time for the teachers and allowing them to gain insight over their entire question set with little effort.

- There are cases in which it is difficult to determine in which class a question should be. For example 'How can the exact position of a triangle vertex be calculated in the Marching Cubes algorithm, when the cell edge on which this vertex must lie is already known? Create a formula for this calculation.' can be considered a question of the category 'applying' but could also be 'creating'. Since our software will be implemented seamlessly in online education platforms it may raise awareness amongst teachers about the importance of defining questions in a way that makes it unambiguous which skills are required to answer them.

A goal of our application is to make it easy for online learning platforms, like FeedbackFruits to implement our software. In order for this service to work our software will consist of the components described in the sections below.

C.2.1. DATA ACQUISITION

In the final product, all sorts of questions are classified. The more diverse the data used to train a classification model, the better the classifier can generalize. In order to achieve good classification we need a large dataset of questions. Our client, feedbackfruits is planning on implementing question generation into their platform. Since automatically classifying these questions as well as teacher generated questions can be very useful to our client we have decided to implement a primitive form of question generation ourselves to expand the data we have available (see subsection C.2.2). To acquire questions we have looked into scraping sources such as *yahoo answers*[1] and *quora* [2], online open courses such as *coursera* [3] and *MIT OpenCourseWare*[4], textbooks and other documents. Unfortunately when researching ways to automatically scrape these sources we ran into the following problems:

- Questions from sources such as *Yahoo* or *Quora* were easy to scrape and have topic information available, but these questions were usually not created for educational purposes and did not provide much value to students.
- Questions from websites such as *MIT OpenCourseWare* were not scrape-able format, as each teacher organizes their course differently.
- Coursera does provide adequate questions in a format that is easy to scape, but does not allow the use of their questions outside of their platform.

We will train our model on 3 datasets that we will create ourselves.

[1] https://answers.yahoo.com/
[2] https://www.quora.com/
[3] https://www.coursera.org/
[4] https://ocw.mit.edu/index.htm

- A dataset of questions from exams and opencourseware scraped and labeled manually
- An edited version of a set of labeled questions found in previous bloom classification research
- A dataset generated by our automatic question generator (see subsection C.2.2)

Our model will be trained on a combination of these 3 datasets. After fully training the model we will compare the classifications it gives on unlabeled data against classifications given by trained educators. This will help us determine what combination of the datasets to use for our model.

C.2.2. QUESTION GENERATION

To train our classifier we would like to use data representative of the data our client will use. FeedbackFruits has a team that is currently working on the automatic generation of questions. In the future FeedbackFruits plans to use the questions resulting from that project as part of the dataset to be classified by our software. Therefore we would like to use our auto-generated questions to see how our software will perform.

C.2.3. PREPROCESSING

It is difficult to predict in advance which preprocessing steps lead to the best results. There are some preprocessing steps that are common for text classification purposes. Our dataset differs from datasets usually used in common text classification approaches in the sense that the samples have no meta data (e.g. a twitter sample has meta-data fields such as 'user', 'location', 'time', 're-tweets', 'likes') and our samples consist of often short single sentence questions. We will try multiple approaches based on previous research. One common approach in text classification is the bag of words or bag of n-grams approach, however this is commonly used on datasets where the samples have multiple sentences whilst ours are often short one sentence questions in which multiplicity of words are uncommon [10]. We will explore this method first.

To improve the accuracy of the model we preprocess the samples in the dataset based on the method described in a paper by Kannan at al. [9]. First we remove stop-words, these are common words (a, the, in, etc.) that don't aid in distinguishing one sentence from the other. According to Zipf's law the frequency of a word is inversely proportional to it's rank [12] which means that words that appear often (such as stop-words) have little added value. Using this observation we can remove stop-words from our questions. The NLTK has a list of stop-words compiled by Porter et al. [16] which we will use.

The same verb conjugated at different tenses can have different spellings and therefore be recognized as different words. By lemmatizing the questions, similar words will be represented by the same feature.

We convert the questions to lowercase to avoid words at the beginning of a questions

becoming a different feature.

When going through a dataset of questions [17] labeled according to their Bloom's taxonomy level we have noticed that the questions labeled as knowledge questions are more often written in the interrogative form (with a question mark at the end of the sentence) than in other categories. The other questions are more often written in the imperative form. We therefore use the presence of an end of sentence punctuation character as an additional feature. Also the order of part of speech tags - such as whether verb subject inversion is present in a sentence - is an important feature to distinguish the form of the question.

We then tokenize the question and use the presence of each unique word as a binary feature in our sample set.

C.2.4. CLASSIFICATION ACCORDING TO BLOOM'S TAXONOMY

Bloom developed a taxonomy that classifies learning objectives into levels of complexity [3]. These learning objectives are tested by asking questions. There are 6 levels of complexity in the taxonomy. Krathwohl et al. [2] later revised the taxonomy, which is the version that we will be using. Each of these levels correspond with examples of typical verbs used in questions are shown in the table below.

Level	Verbs associated
Remember	Select, List, Name, Define, Describe, Memorize, Label, Identify, Locate, Recite, State, Recognize
Understand	Match, Restate, Paraphrase, Rewrite, Give Examples, Express, Illustrate, Explain, Defend, Distinguish, Summarize, Interrelate, Interpret, Extend
Apply	Organize, Generalize, Dramatize, Prepare, Produce, Choose, Sketch, Apply, Solve, Draw, Show, Paint
Analyze	Compare, Analyze, Classify, Point Out, Distinguish, Categorize, Differentiate, Subdivide, Infer, Survey, Select, Prioritize
Evaluate	Judge, Relate, Weight, Criticize, Support, Evaluate, Consider, Critique, Recommend, Summarize, Appraise, Compare
Create	Compose, Originate, Hypothesize, Develop, Design, Combine, Construct, Produce, Plan, Create, Invent, Organize

One option is to use these words as binary features. Not every question will contain one of the above mentioned verbs however. Other features such as the part of speech (POS) tagging will also be extracted from the questions.

SVM's, Naïve Bayes and k-nearest neighbour classifiers are often used for text classification purposes. A majority vote of the outputs of these machine learning algorithms

could also be an option for classification.

C.2.5. ACTIVE AND ONLINE LEARNING

The software will make use of active to speed up the labeling of the dataset and online learning techniques to make use of user feedback about the results instantly to improve accuracy. The UML diagram in Figure C.1 shows the high level relationship between the classifier, the datasets and the active and online learning components. Active learning will propose the least certain samples to be labeled next, till an accuracy threshold is met. This will reduce our client's labeling costs for new data sets. When our client deploys their software system at a new educational institution it is possible that the bloom's taxonomy classifier encounters question forms it has not seen before (i.e. due to professors of new courses making use of the software). Online learning will allow the model to be 'recalibrated' quickly so that the classifier can respond to new data faster than running a batch process to retrain the model.

LABELING OUR DATASET

The dataset will consist of labeled and unlabeled questions. We use supervised machine learning algorithms to train our classifier. The more labeled data we have, the better our classifier performs. When expanding the dataset of questions, our client might want to label all or a part of the new samples. By means of active learning we select the samples of the unlabeled dataset - that the classifier is most uncertain about - to label. By doing so we improve the model with the less labeling effort compared to labeling all samples.

INCORPORATING USER FEEDBACK

Not every label given by the classifier will be correct. We give teachers the option to give incorrect labeled questions the correct label. We want to prevent the entire model to be retrained every time there is a new correct label because it is computationally demanding. Therefore we will define a batch size of new corrects labels and implement an out of core method of learning.

C.3. USABILITY

An essential component of our final product is a Bloom's taxonomy classification tool. We are going to showcase this tool by making a GUI in which users enter a question and can see the predicted bloom's taxonomy class.

The GUI will contain a section where a question can be entered and a results section. After a user enters a query he will be taken to the results section where the prediction results with the confidence level per taxonomy class are displayed.

The results section allows the users to give feedback, either agreeing or disagreeing with the prediction. When users disagree they are given the option to select the class they think the question belongs to. There will also be a subsection where the users have the option to view suggested questions. Feedback on the predicted class of the user query and of the suggested questions will be used to update the online learning system.

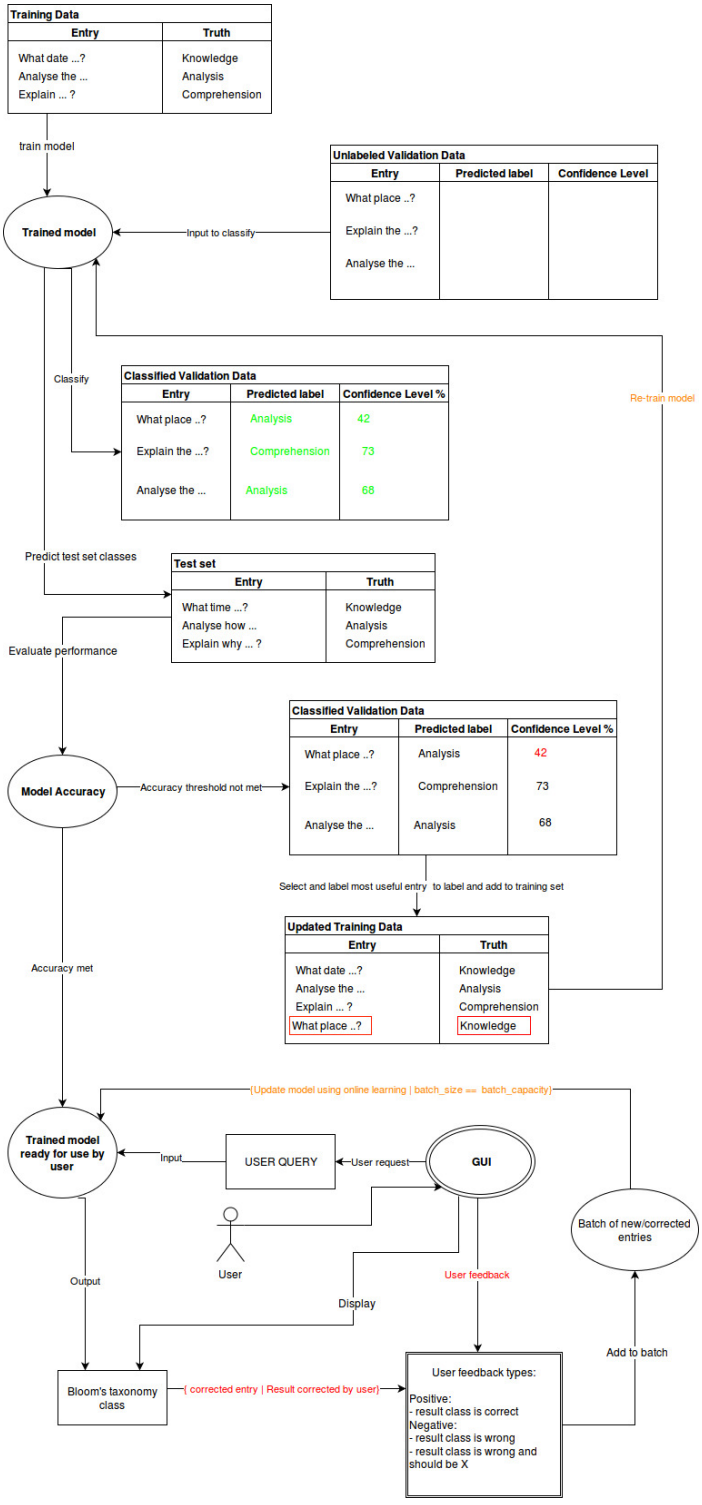


Figure C.1: The way our model handles data and learns from new information.

The GUI must be user friendly. Entering questions and getting predictions should require few actions from the user considering a single user might need to perform a large amount of question classifications in a row (e.g. when entering all questions of a sample exam). For the feedback system to be used by users it must not feel like a burden. Therefore the GUI will have standard buttons that allow users to give feedback (e.g. "correct prediction", "incorrect prediction", "result should be class X") in one click.

C.4. DEVELOPMENT TOOLS AND LIBRARIES

C.4.1. CHOICE OF PROGRAMMING LANGUAGE

We have chosen to use Python as our programming language for the project. Many libraries and tools we would like to use for the project are written in Python. Also Python is often used for data analysis and machine learning software. Relevant documentation can be found online along with tutorials and solutions to common problems.

C.4.2. POSTGRESQL

In our choice of database we considered both a relational database and a document based database. While document based database like MongoDB can be easier to set up since it doesn't require a structure to be defined beforehand. We decided to go with the relational database PostgreSQL instead. Considering that the software is going to be used in an online learning environment where data will get added continuously, it is safe to assume our data will get very large. Depending on how it will be used the feedback of teachers on the classification of their questions consists of relational data that can be better represented in a relational database. The use of PostgreSQL over MySQL was decided to accommodate the current data structure of the client.

C.4.3. SCI-KIT LEARN

We chose to use Scikit-learn as a library because it offers high-level implementations of many common machine learning algorithms. The machine learning algorithms we are planning to use are included in this library. The Scikit-learn library is also very well documented. `Scikit-Learn.org` offers a wide range of examples of the functionalities.

C.4.4. NATURAL LANGUAGE TOOLKIT (NLTK)

The NLTK is a suite of libraries containing many tools that are used in natural language processing. We will use the NLTK for text-preprocessing purposes it has all the functionalities that we need for this.

C.5. KNOWN SIMILAR PROBLEMS

There has been some research in the field of question classification according to Bloom. The research of Yahya et al. [17] focuses on using SVM's for this task. The precision and accuracy look promising (87.4% and 85.33% respectively), however their dataset is small and almost every sample explicitly begins or contains one of the keywords listed in table with associated verbs. Since most teachers are trained to use keep some form of blooms taxonomy in mind[1] these keywords often appear in exam and practice questions,

however not all exam questions contain these keywords. They also only use the term frequency and inverse document frequency as features.

In the research of Zhang et al. [6] only looks a keywords on a dataset that is more representative of real world questions. Without using a machine learning approach they managed to get the correct answer 75% of the time for knowledge questions. Their success rate other classifications is very low however, with evaluation questions only guessed correctly 25% of times. The average success rate of all classifications is 47%.
Osadi et al.[15] combine multiple classifiers (rule based, support vector machine, k-nearest neighbor and Naive Bayes) by means of majority voting. Their dataset however consisted of 100 questions. Since the dataset is relatively small for a machine learning algorithm they have decided to improve on this data in further research to gain more conclusive results.

Automatic question generation (AQG) is a complicated subject because it requires natural language understanding (NLU) and natural language generation (NLG) to work well. NLU is an AI-complete problem [8] which implies that computers need to have the same level of understanding as a human to solve these problems - it is far out of our reach and a smart approach is needed to get relevant results. AQG is often approached using templates with placeholders and lists of fill-in words to fill the templates [5]. Using NLU techniques to generate questions from templates can give better results than previous methods as demonstrated by [14] and [11]. Extracting understanding from sentence structure and choosing question templates that match the knowledge extracted from the sentence is amongst the most promising techniques for AQG [13].

C.6. CONCLUSION

After researching the subject we conclude that building a good Bloom classifier is possible. We will explore the best combination of features and machine learning classifiers. Naïve Bayes and SVM's are the two most promising machine learning algorithms. We cannot predict however which machine learning algorithm will perform the best. Due to the short nature of our text data we will have to explore both options that use a bag of words approach and one that has the presence of words as binary feature.

We think it will prove to be difficult to generate high quality questions that span the entire Bloom taxonomy. Most research conducted in this area generate questions that mostly fall in the knowledge class of Bloom's taxonomy. We haven't come across any general templates that can be used for questions higher in the taxonomy, but we are convinced it is possible. Our goal is to make specific templates to generate questions of every level of Bloom's taxonomy.

REFERENCES

[1] Nancy E Adams. Bloom's taxonomy of cognitive learning objectives. *Journal of the Medical Library Association : JMLA*, 103(3):152–153, jul 2015. doi: 10.3163/1536-5050.103.3.010. URL https://doi.org/10.3163/1536-5050.103.3.010.

[2] Lorin W Anderson. An empirical investigation of individual differences in time to learn. *Journal of educational psychology*, 68(2):226, 1976.

[3] Benjamin S Bloom et al. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay*, pages 20–24, 1956.

[4] Phyllis Blumberg. Maximizing learning through course alignment and experience with different types of knowledge. *Innovative Higher Education*, 34(2):93–103, 2009.

[5] Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. Automatic question generation for vocabulary assessment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 819–826, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220678. URL http://dx.doi.org/10.3115/1220575.1220678.

[6] Wen-Chih Chang and Ming-Shun Chung. Automatic applying bloom's taxonomy to classify and analysis the cognition level of english question items. In *Pervasive Computing (JCPC), 2009 Joint Conferences on*, pages 727–734. IEEE, 2009.

[7] Meredith D. Gall. The use of questions in teaching. *Review of Educational Research*, 40(5):707–721, 1970. ISSN 00346543, 19351046. URL http://www.jstor.org/stable/1169463.

[8] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: The state of the art. *Comput. Linguist.*, 24(1):2–40, March 1998. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972719.972721.

[9] S Kannan and Vairaprakash Gurusamy. Preprocessing techniques for text mining. 2014.

[10] Anthony Khoo, Yuval Marom, and David Albrecht. Experiments with sentence classification. In Lawrence Cavedon and Ingrid Zukerman, editors, *Proceedings of the 2006 Australasian language technology workshop*, pages 18–25, 2006.

[11] Ming Liu, Rafael A Calvo, Anindito Aditomo, and Luiz Augusto Pizzato. Using wikipedia and conceptual graph structures to generate questions for academic writing support. *IEEE Transactions on Learning Technologies*, 5(3):251–263, 2012.

[12] Christopher D Manning, Hinrich Schütze, et al. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[13] Karen Mazidi and Paul Tarau. Automatic question generation: From nlu to nlg. In *Proceedings of the 13th International Conference on Intelligent Tutoring Systems - Volume 9684*, ITS 2016, pages 23–33, New York, NY, USA, 2016. Springer-Verlag New York, Inc. ISBN 978-3-319-39582-1. doi: 10.1007/978-3-319-39583-8_3. URL https://doi.org/10.1007/978-3-319-39583-8_3.

[14] Assel Omarbekova, Altynbek Sharipbay, and Alibek Barlybaev. Generation of test questions from rdf files using python and sparql. In *Journal of Physics: Conference Series*, volume 806, page 012009. IOP Publishing, 2017.

[15] K. A. Osadi, M. G. N. A. S. Fernando, and W. V. Welgama. Ensemble classifier based approach for classification of examination questions into bloom's taxonomy cognitive levels. *International Journal of Computer Applications*, 162(4):1–6, Mar 2017. ISSN 0975-8887. doi: 10.5120/ijca2017913328. URL http://www.ijcaonline.org/archives/volume162/number4/27228-2017913328.

[16] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[17] Anwar Ali Yahya, Addin Osama, et al. Automatic classification of questions into bloom's cognitive levels using support vector machines. 2011.