Automatic programming problem difficulty evaluation – first results

Artūras Skarbalius[1] and Mantas Lukoševičius[1[0000−0001−7963−285X]]

Faculty of Informatics, Kaunas University of Technology, Kaunas, Lithuania
{arturas.skarbalius,mantas.lukosevicius}@ktu.edu

Abstract. In this work, we address automatic evaluation of the difficulty of programming problems or exercises. Typically the problems consist of both text description and accompanying figures. We collect a suitable dataset, investigate the evaluation based on the text and the image data separately, as well as a combination of the two. The first results of this investigation are reported, together with the discussion and future work.

Keywords: Programming problem · Difficulty evaluation · Natural language processing · Convolutional neural network · Deep learning · Machine learning.

1 Introduction

Programming problems of various kinds are regularly used to teach those interested in programming how to do so effectively in a practical way, and on occasion, they are also utilized to evaluate or demonstrate the ability of an individual regarding a particular field.

The issue with these sorts of problems, however, is that compared to other methods, these kinds of programming problems tend to be comparatively subjective - there is oftentimes very little objective way to evaluate exactly how difficult an exercise would be due to those writing them having a different perception of it compared to everyone else.

This project is an attempt to resolve part of this issue by utilizing machine learning to evaluate difficulty in regards to programming problems. Utilizing this system, it should be possible to have a more objective view of a problem and make it easier to perform the aforementioned teaching and evaluation of skills if successful.

The problem formulation usually includes both text and images, which makes this task more difficult, since typically different machine learning methods are used for the two types of data. Here we attempt to investigate the difficulty estimation based both on text, rendered image of the problem, and combination of both.

We review the previous related work in Section 2, explain the data that we use in Section 3, present our methods used and preliminary results in Section 4, and finally, a short discussion is given, alongside ways to improve it in the future, in Section 5.

2 Related Work

Here we review the basic approaches and algorithms used for this type of problem. Due to the unique nature of the task, exact methods for evaluation have not been fully considered. As such, a wider variety of methods is considered.

2.1 Approaches

There have not been many attempts to approach this problem in particular, but there have been attempts to achieve results in similar fields. One example in particular is an attempt at predicting the difficulty of various questions in reading problems [5] that utilized a particular framework called the Test-aware Attention-based Convolutional Neural Network (TACNN). According to the given results in the paper, the approach does improve on the result by some amount, and experts generally have lower accuracy on the evaluation than the resulting neural network.

Another notable approach has been the utilization of a hybrid AI [7] to evaluate exercise difficulty. To initialize the system, a teacher, as an expert, needs to input a rule set. After this is done, the feedback of any students that have taken the exercise is taken and used to adjust the results via a genetic algorithm. The results obtained from such indicates that a significant portion of exercises are initially evaluated incorrectly. However, this approach appears to be flawed - the method used indicates exactly one set of rules per difficulty level, which is fairly inaccurate to a realistic situation. In addition, if multiple rules for a single difficulty level are included in the starting rule set, the system cannot correctly determine which to use as the starting point to adjust from. According to the article, the rule to be used as a base for the genetic algorithm to adjust was chosen at random at the time of writing.

Beyond this, there do not appear to be many other methods utilized for tasks of this nature.

2.2 Algorithms

There are numerous algorithms that can be utilized. As the method we have been using (detailed in the Methods section) involves both text and images, there will be two subsections for algorithms associated with both, as well as another subsection indicating ways to combine multiple methods into one.

Text-Based Evaluation Natural Language Processing (NLP) initially requires some amount of pre-processing for the text [10]. There are multiple parts of pre-processing that can be utilized, as well as many different methods, so only a small portion will be mentioned here:

1. **Tokenization** is a process during which large amounts of text are separated and occasionally classified into smaller sections that machine learning can utilize effectively. Commonly used methods involve separating text by sentences or separating text by words.

2. **Normalization** is a method utilized to improve a system by attempting to remove redundant words with similar meanings. **Stemming**, for instance, is a method used to reduce words to their root form, and **lemmatization** removes prefixes and suffixes from words.

As an example, the **bag-of-words** method gathers all of the tokens obtained from a text into a "bag" – a format that completely ignores word order and grammar, but retains the number of times each word has shown up in a text in a way that a computer can use in the future.

For text-based evaluation, various algorithms can be used. To simplify things, just some of the useful methods that have been evaluated will be described here.

1. **A decision tree** [8] is a simple method for classification. It makes use of multiple true/false statements to form a result. A decision tree classifier in machine learning forms this sort of tree through the use of training data to evaluate the exact values necessary. It is commonly used for relative simplicity and effectiveness, and there are several methods to improve the accuracy of these kinds of methods as well (detailed further under Model combination). A simple example of the way a decision tree works is shown in Figure 1 – In this case, Y is a binary variable, while the other two variables, X1 and X2, are used in this case to determine what the result should be.
   Once a model is formed, pruning can be performed for the sake of lowering the redundant or otherwise excessive branches and simplify the resulting model. While unnecessary, it can help with speed on several occasions.

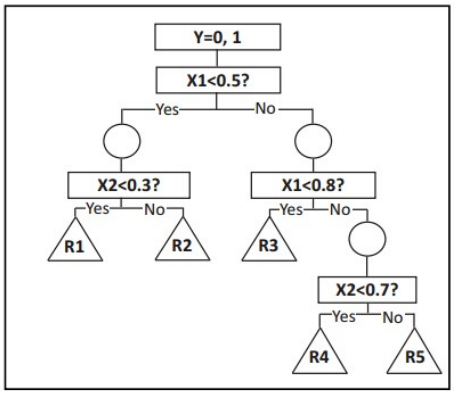


Fig. 1. Sample decision tree based on binary target variable Y. The image is used from [8] under the Creative Commons Attribution-NonCommercial-Share Alike 4.0 Unported License.

2. Light Gradient Boosting Machine (**LightGBM**) [6] is a gradient boosting decision tree (GBDT), designed by Microsoft. In particular, this implementation adds a couple of techniques: Gradient-based One-Side Sampling

(GOSS), which allows for an accurate information gain with less data, and Exclusive Feature Bundling (EFB), which bundles mutually exclusive features together in a way that would allow for fewer variables that the model has to look into. Through this, the resulting method is approximately 20 times faster while providing roughly equivalent accuracy.
3. The Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (**L-BFGS**) [1], is a method to approximate the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm using a limited amount of available memory. Oftentimes this method is used for parameter estimation. The algorithm estimates an inverse Hessian matrix to help with determining the exact value.

Some classification methods are only created for the purposes of binary classification. To change these into multiclass classification, there are two methods to go about this, in particular [3]:

1. **One-vs-All** method creates n different versions of the same model, where n is the number of different classifications available, with each one having two different possible results - it is one of the available classifiers, or it is any of the others. When evaluating, the resulting model returns the classifier for which the model had the highest result. This method is generally faster and requires less memory to utilize.
2. **One-vs-One** method creates a different model for each possible combination of classifiers available, with each one having two of the possible multiple classifiers available. Once all of the different models finish, the correct answer is chosen by summing up or averaging all scores, and returning the highest result. This method is more complicated, slower, and requires more memory to utilize, but it may obtain more accuracy on occasion compared to One-vs-All.

Image-based Evaluation Computer vision is, to put it simply, an attempt to make a computer interpret vision in a similar way any other individual does [11]. Thus far, most cases of computer vision involve extremely specialized forms, such as **optical character recognition** (OCR) to interpret symbols from images. While more general-purpose methods do exist, they are not quite applicable to the task at hand presently, though there is a good chance that it will be more possible to use in the future.

For image-based evaluation, the main thing that is used is neural networks - a network made of several layers of interconnected artificial neurons that take in a value and return a result based on that value. The most common form of these neural networks that are utilized are **convolutional neural networks** [9]. Various sorts of variations and advancements have been made over the years, but as a general case, these sorts of neural networks rely on three forms of neuron layers:

1. **Fully-connected layers** are regularly used in all forms of artificial neural networks. The main characteristic of these layers is that they are completely

connected to all adjacent layers, without being connected in any way to any other layers.
2. **Convolutional layers**, which are capable of learning various kernels that are then utilized to more efficiently evaluate a given image, are the primary layers the convolutional neural network utilizes. In most standard neural networks, reading an image would regularly result in a model that becomes too large to train in any effective manner. By using a convolutional layer, it is possible to reduce the complexity of a model significantly.
3. **Pooling layers**, which aim to pool together values in an attempt to gradually reduce the dimensions of the received image in such a way that it could be reliably used without making the system too complex.

Model Combination Several different methods have been used to put together multiple models in a way that would make the end result more capable than any of the individual parts. Oftentimes, these models are mostly identical to the others it is being combined with, excluding the data it was trained with. This sort of model combination is referred to as ensemble learning. The most common examples of this are [12]:

1. **Bootstrap aggregating**, sometimes referred to as **bagging**, is by far one of the simplest methods of ensemble learning available. For this method, the data is randomized during learning for each of the base learners. The results are counted for each of the available results, then these results are taken and averaged. The value that has achieved the highest average is considered to be the correct result. One example of this is random forests - an ensemble model that makes use of many different decision trees to form a result, regularly more accurate than any of the individual decision trees. It is a particularly quick to create model, but it is significantly less accurate compared to the others.
2. **Bayesian model averaging** [4] is created in a fairly similar manner to bootstrap aggregating - the data is randomized during learning, then an overarching model puts together the result. The key difference, however, is that each of the individual models in this exact scenario has a set weight during the evaluation, influencing the end result in this way. The accuracy is significantly better compared to Bootstrap aggregating due to this.

Aside from this, there is another method to utilize multiple different models, however this one requires more specific methods - **two-branch neural networks**. [2] These networks utilize two different models - for example, one that takes in text and another that takes in images, obtains the partial results, then runs those results through another layer of neurons to get a final result. By making use of this, both text and images are evaluated together.

3 Datasets

The dataset has been created personally through web scraping from several free websites that can be utilized, such as hackerrank.com and codechef.com. The

finalized dataset contains over 5000 different entries, each one containing difficulty (provided by the author), accuracy (a percentage of how many individuals successfully provided a suitable answer to the problem), the problem name, a description, and a variable number of images. At present, the model attempts to make use of the provided difficulty for the evaluation.

After this is finished, the data is recreated into multiple other methods to ensure simplicity for the actual evaluation. A CSV file is created from the text data (difficulty, problem name, description), and image data is combined with the description to create a set of images that contain the necessary text for evaluation.

4 Methods and Results

The methods we have utilized involve several different text classification methods to obtain different results - perceptrons, linear support vector machines, etc. A two-branch neural network is also used to improve the evaluation by adding deep-learning-based image recognition. A specially-made program is used to obtain results and return the possible accuracy.

4.1 Text-Based Evaluation

The results for text evaluation are fully detailed in Table 1. In this case, "MicroAccuracy" refers to an accuracy calculation from all available results, while "MacroAccuracy" takes the precision and recall in place of it instead.

The text preprocessing used for each of the models is as follows: Each of the possible end result labels is assigned a numerical key, then all of the text from the other two possible inputs is transformed into a vector of floats representing counts of n-grams (an identical continuous sequence of n items) for both words and characters. Through doing this, it can then evaluate what each word indicates for a problem's difficulty.

For the sake of a more comprehensible explanation, the basic way accuracy is calculated is

$$P_r = \frac{TP}{TP + FP}, \tag{1}$$

where TP refers to the number of true positives, and FP refers to the number of false positives.

Micro-average Accuracy, written here as MicroAccuracy, refers to adding all of the values together before calculations. So, as an example, if there were three classes to evaluate from, micro-average accuracy would be counted as

$$P_{r_{micro}} = \frac{TP1 + TP2 + TP3}{TP1 + TP2 + TP3 + FP1 + FP2 + FP3}. \tag{2}$$

Macro-average accuracy, written here as MacroAccuracy, refers to dividing each of the results before averaging them, thus ensuring that each class has an

equal contribution to the result. Using the same example of three classes, this is how macro-average accuracy is calculated as

$$P_{r_{macro}} = \frac{\frac{TP1}{TP1+FP1} + \frac{TP2}{TP2+FP2} + \frac{TP3}{TP3+FP3}}{3}. \quad (3)$$

Micro-average accuracy is preferable when trying to calculate accuracy for a multi-class classification problem if it is believed that there may be some imbalance between the number of entries per class. In this case, macro-average accuracy is used to show that, while there is a small amount of imbalance in the data, it does not truly impact the result with most models.

Table 1. Text evaluation results

Model name	MicroAccuracy	MacroAccuracy
Random Forest	0.718	0.725
Decision Tree	0.711	0.712
LightGBM	0.701	0.708
Logistic Regression (L-BFGS)	0.660	0.657
Maximum Entropy (SDCA)	0.639	0.661
Stochastic Gradient Descent	0.620	0.623
Averaged Perceptron	0.609	0.641
Linear Support-Vector Machine	0.583	0.597
SymbolicSgdLogisticRegression	0.545	0.584
Maximum Entropy (L-BFGS)	0.458	0.319

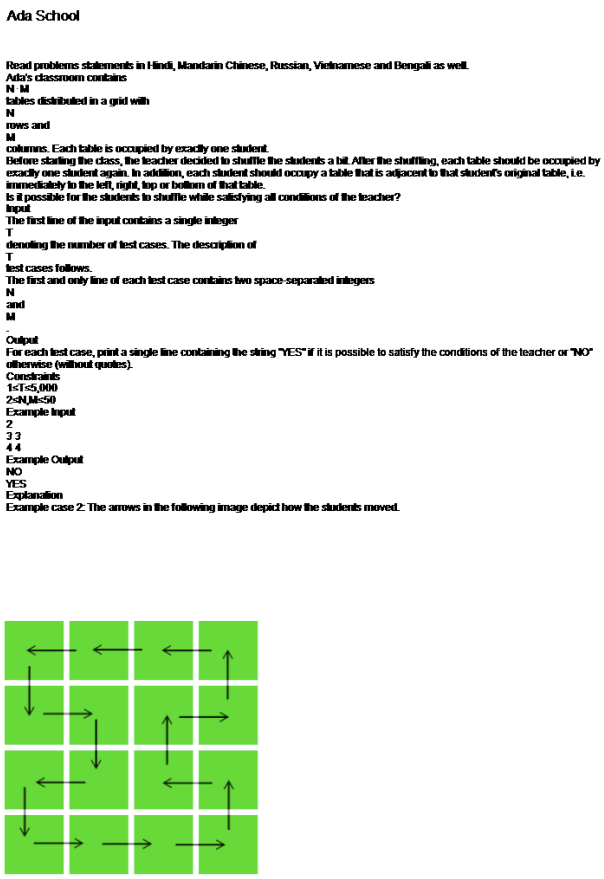
The algorithms mentioned use the same preprocessing methods, and no method of separating parts of the exercise are utilized. Regardless, limited manual testing shows that the results do not vary significantly if minor or irrelevant parts are changed.

4.2 Image-Based Evaluation

For image-based evaluation, a single image containing the whole problem description together with the illustrations was rendered for each problem. The size needed for the image is pre-calculated, the text and images are taken, then the text is rendered using a predefined style, with images being inserted in places where they were in the original problem. An example of the image formed from the text and images of an exercise is shown in Figure 2.

Once that is finished, the result is used as data for a deep neural network, by which the data is evaluated. Several well known models were used – their names as well as the obtained results are shown in Table 2.

Similarly to the text models, preprocessing is the same for each model – in this case, simply loading the image as raw bytes into the model. The rest is handled by the neural network for each model.


Ada School

Read problems statements in Hindi, Mandarin Chinese, Russian, Vietnamese and Bengali as well.
Ada's classroom contains
N · M
tables distributed in a grid with
N
rows and
M
columns. Each table is occupied by exactly one student.
Before starting the class, the teacher decided to shuffle the students a bit. After the shuffling, each table should be occupied by exactly one student again. In addition, each student should occupy a table that is adjacent to that student's original table, i.e. immediately to the left, right, top or bottom of that table.
Is it possible for the students to shuffle while satisfying all conditions of the teacher?
Input
The first line of the input contains a single integer
T
denoting the number of test cases. The description of
T
test cases follows.
The first and only line of each test case contains two space-separated integers
N
and
M
.
Output
For each test case, print a single line containing the string "YES" if it is possible to satisfy the conditions of the teacher or "NO" otherwise (without quotes).
Constraints
1≤T≤5,000
2≤N,M≤50
Example Input
2
3 3
4 4
Example Output
NO
YES
Explanation
Example case 2: The arrows in the following image depict how the students moved.

Fig. 2. Example of an image formed from a problem's text and images. The problem used is from codechef.com.

Table 2. Image evaluation results

Model name	MicroAccuracy	MacroAccuracy
ResNet50	0.403	0.343
ResNet101	0.436	0.385
MobileNet V2	0.297	0.239
Inception V3	0.339	0.258

4.3 Combining Text and Image-Based Evaluation

In an attempt to improve the accuracy of the text-based and image-based methods, a two-branch neural network was used. The two separate models are used initially – one for text, the other for images. The outputs of the two models are

then concatenated and re-evaluated to obtain the refined result. The two-branch neural network we are using is presented in Figure 3. For now, we use a single layer neural network for the results refinement part.

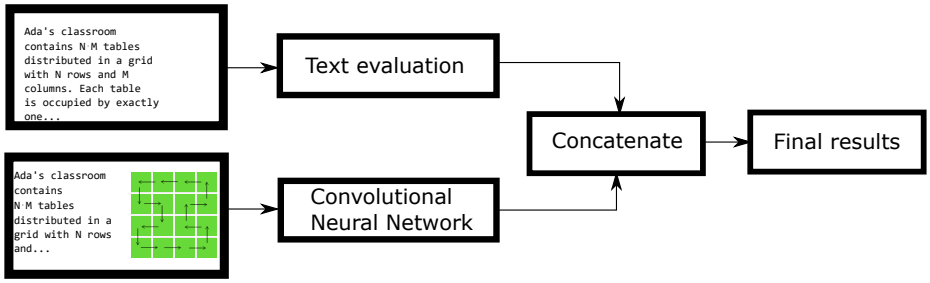


Fig. 3. Example of a two-branch neural network being used.

Before using the two-branch neural network, our image classification accuracy was about 40-50 %. When a two-branch neural network is used, the accuracy goes up rapidly, up to about 80+ % – significantly higher than either of the individual models.

Table 3. Multiple variations for two-branch neural networks

Image model name	Text model name	MicroAccuracy	MacroAccuracy
ResNet50	Random Forest	0.967	0.969
ResNet50	Logistic Regression (L-BFGS)	0.925	0.937
ResNet50	Averaged Perceptron	0.857	0.839
ResNet50	Stochastic Gradient Descent	0.804	0.793

5 Discussion and Future Work

At the time of writing, the things that have been attempted thus far are as follows:

1. Text-based processing
2. Image-based processing
3. Two-branch neural networks

With this, a varied set of initial results has been obtained, giving an opportunity to evaluate some of the given results and potentially improve on them.

Among text-based models, with a bag of n-grams preprocessing, the Random Forest ensemble model returned the best results.

Future work for text-based models may include the use of different text preprocessing methods – the completely different structure of the input may increase the accuracy of the models further.

More options for the image recognition network could be tested in the future.

When fully trained, the models can most likely accurately determine the difficulty of exercises correctly, so long as the data used for training is focused on the type of exercise being evaluated by the model. It is unclear whether or not these models can function for a wider variety of exercises simultaneously without additional adjustments, as no testing has been done in regards to this.

The two-branch neural network is particularly useful at the moment. The accuracy of the two-branch neural network is significantly greater (80 %) than either the text-based (72 %) or image-based (43 %) models on their own.

More options for the combination of the two branches could be considered in the future, including a single fully trainable architecture with no intermediate interpretable results from the text and image-based models.

References

1. Bollapragada, R., Nocedal, J., Mudigere, D., Shi, H.J., Tang, P.T.P.: A progressive batching L-BFGS method for machine learning. In: International Conference on Machine Learning. pp. 620–629. PMLR (2018)
2. Chen, H., Lagadec, B., Bremond, F.: Partition and reunion: A two-branch neural network for vehicle re-identification. In: CVPR Workshops. pp. 184–192 (2019)
3. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., Herrera, F.: An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. Pattern Recognition **44**(8), 1761–1776 (2011)
4. Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging: a tutorial. Statistical science pp. 382–401 (1999)
5. Huang, Z., Liu, Q., Chen, E., Zhao, H., Gao, M., Wei, S., Su, Y., Hu, G.: Question difficulty prediction for reading problems in standard tests. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
6. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30**, 3146–3154 (2017)
7. Koutsojannis, C., Beligiannis, G., Hatzilygeroudis, I., Papavlasopoulos, C., Prentzas, J.: Using a hybrid AI approach for exercise difficulty level adaptation. International Journal of Continuing Engineering Education and Life Long Learning **17**(4-5), 256–272 (2007)
8. Myles, A.J., Feudale, R.N., Liu, Y., Woody, N.A., Brown, S.D.: An introduction to decision tree modeling. Journal of Chemometrics: A Journal of the Chemometrics Society **18**(6), 275–285 (2004)
9. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458 (2015)
10. Palmer, D.D.: Text preprocessing. Handbook of natural language processing **2**, 9–30 (2010)
11. Szeliski, R.: Computer vision: algorithms and applications. Springer Science & Business Media (2010)
12. Zhou, Z.H.: Ensemble learning. Encyclopedia of biometrics **1**, 270–273 (2009)