Few shot learning approaches to essay scoring

Robert K. Helmeczi†, Savas Yıldırım†, Mucahit Cevik†*, Sojin Lee‡
† Toronto Metropolitan University, Toronto, Canada
‡ Blees AI, Toronto, Canada

Abstract
Automated essay scoring (AES) involves using computer technology to grade written assessments and assigning a score based on their perceived quality. AES has been among the most significant Natural Language Processing (NLP) applications due to its educational and commercial value. Similar to many other NLP tasks, training a model for AES typically involves acquiring a substantial amount of labeled data specific to the essay being graded. This usually incurs a substantial cost. In this study, we consider two recent few-shot learning methods to enhance the predictive performance of machine learning methods for AES tasks. Specifically, we experiment with a prompt-based few-shot learning method, pattern exploiting training (PET), and a prompt-free few-shot learning strategy, SetFit, and compare these against vanilla fine-tuning. Our numerical study shows that PET can provide substantial performance gains over other methods, and it can effectively boost performance when access to labeled data is limited. On the other hand, PET is found to be the most computationally expensive few-shot learning method considered, while SetFit is the fastest method among the approaches.

Keywords: Automated essay scoring, NLP, few-shot learning, transformers

1. Introduction

Automatic Essay Scoring (AES) is defined as the task of automatically grading written assignments. This task has been conceptualized as early as 1966, when Page [1] showed that a computer could produce essay judgements that were nearly indistinguishable from those of a set of domain experts. AES encompasses a wide array of assessments: standardized examinations, long answer questions, and high school essays are just some of the evaluations that can be addressed with automatic essay scoring. The grading for each of these assessments can vary drastically: while some graders will put considerable weight on how well an essay conforms to linguistic rules, the marker for a long answer question might just be assessing whether or not the student addresses a few key concepts.

![Figure 1. Automatic essay scoring framework considered in our study. 1] The essay is split into individual sentences. 2] Each sentence is assigned one or more labels. 3] The labeled sentences are graded and a final score is assigned.]

In this research, we consider an AES problem which attempts to mark a student’s essay based on their understanding of core ideas, verifying whether or not they were addressed by the student. The procedure for the considered AES framework is outlined in Figure 1, where, notably, each sentence is considered individually. The focus of this research is on training the labeler/classifier. This process is complicated by the fact that some sentences can address multiple concepts at once. Accordingly, the dataset is multi-labeled, allowing for each sentence to fall under one or more classes. With the advent of pre-trained language models, the ability to train a predictor to grade essays in this manner has been enhanced considerably. However, these

*mcveik@torontomu.ca
pre-trained models frequently require access to a substantial amount of hand-labeled data in order to achieve satisfactory results. To address this issue, we consider few-shot learning, the field of artificial intelligence in which models are trained with access to limited training data. We consider two popular few-shot learning approaches for text classification, namely, pattern-exploiting training (PET) and SetFit, and adapt them to our AES task.

Each of the approaches considered in this work relies on pre-trained language models, but the mechanism in which they were applied differs. PET reformulates an input sentence into a cloze-style phrase in order to add context to a given task, and uses masked-language modeling (MLM) to predict labels [2, 3]. SetFit leverages the sentence embeddings used by pre-trained language models to increase the size of the training data without requiring additional labeled examples [4].

The objective of this research is to investigate the few-shot performance of state-of-the-art models on our AES task. The contributions of our study can be summarized as follows:

- We adapted several popular few-shot learning strategies to a novel AES problem. This process includes designing custom-made patterns and verbalizers for the PET algorithm.
- We showed the effectiveness of multi-label few-shot learning in the context of the AES problem.
- We provided a detailed numerical study on few-shot learning approaches in a wide variety of contexts, from single-label classification with two labels to multi-label classification with 51 labels. Hence, our analysis helps establish the strengths and weaknesses of the approaches considered, both in terms of data cardinality and label types.

2. Literature Review

While high-quality datasets can sometimes be purchased, in instances where task-specific data is necessary, it is often the case that relevant labeled datasets do not exist. In these scenarios, labeled data must be manually created, often through the use of domain experts, which incurs a substantial cost. Recently, few-shot and zero-shot learning approaches have become popular, as they often allow for training satisfactory models with access to just a handful of labeled training instances. The key to the success of these approaches often involves leveraging the power of pre-trained language models.

There exist various strategies for handling zero- and few-shot learning. In-context learning (ICL) is one such approach; it does not require any supervised learning, and a model can be applied to multiple tasks without the need for fine-tuning or parameter updates. The ICL inference process typically involves passing an input directly to the language model alongside some training samples, known as shots. An exemplar of this technique is the GPT model, as outlined by Brown et al. [5]. The main drawbacks of GPT models are their size—GPT-3 has 175 billion parameters—and the inference complexity caused by passing multiple examples to the model at once.

The computational cost of GPT makes a task pipeline impractical, so parameter-efficient prompt-based approaches have been proposed to address the need for smaller language models that can work well with limited labeled data [6–8]. In parameter-efficient approaches, only some of the parameters are trained for a downstream task while the rest are frozen. This helps reduce the number of parameters needed while still achieving the desired results. While vanilla fine-tuning updates all parameters of the pre-trained model, it is also possible to train a relatively small number of parameters, as proposed in the concept of adapters [9]. T-Few [10] is another parameter efficient approach for few-shot learning that achieved good results on many Natural Language Processing (NLP) benchmarks. It employs a novel technique called \((LA)^3\) to update a limited number of parameters during training. The adapters insert small, trainable feed-forward networks between the layers of the pre-trained model. Two examples of such models are prompt-tuning, and prefix-tuning [11, 12], which can efficiently train the model parameters, adding trainable continuous embeddings (also called continuous prompts). They combine learned continuous embeddings of prompts with the input to complete a task. Without fine-tuning all of the parameters of the model, they are still able to match its performance by updating only a small fraction.

Several studies have examined few-shot fine-tuning with discrete prompts, where no learned prompt is used, but instead, concrete words are employed. PET [2, 3] is a prompt-based technique that has gained notable attention. This algorithm relies on reformulating input examples as cloze-style questions and using
masked language modeling to leverage domain expertise for few-shot learning [2, 3]. With access to just 32 labeled examples, PET was able to achieve superior performance to GPT-3 with models that are magnitudes smaller in size [3]. PET utilizes task-specific unlabeled data to generate additional labeled data points.

The main limitation of PET-like algorithms is that they rely on manually-crafted prompts and require task-specific unlabeled datasets. ADAPET provides a solution to the need for unlabeled datasets in the PET pipeline [13]. It is built on top of PET, but focuses on few-shot learning without any unlabeled data, unlike PET. ADAPET modifies the objective of PET to provide more dense supervision during fine-tuning.

A different line of research solely focuses on zero-shot learning. These models are trained on a large scale to succeed in tasks unseen before. The most impressive example is the T0 model [14], which outperforms baseline models in certain zero-shot learning tasks, showing its potential for real-world applications. These approaches use transfer learning, meta-learning, and self-supervised learning to create models that can quickly adapt to new tasks.

Other approaches rely solely on contrastive learning to artificially increase the number of training data points and enhance model capacity. A recent successful adaptation of these methods in NLP is SetFit [4]. This method applies a contrastive fine-tuning approach to improve few-shot scenarios. It creates a contrastive binary-labeled dataset by grouping sentences with the same label, which are assigned a positive label in the contrastive data set; and different labels, which are assigned a negative label in the contrastive dataset.

Automated essay scoring has been studied extensively in the literature starting in the 1960s [1]. Due to the subjectivity of some AES tasks, many early AES models required not only a substantial amount of labeled data in the form of marked essays, but they also required essays to be marked by a large number of human graders [15]. These models often picked key features for statistical modeling to score essays [15]. Ke and Ng [16] provided a detailed literature review on essay scoring. They identified the dimensions of essay quality (e.g., grammar, relevance, development, and thesis clarity), and evaluated popularly used corpora (e.g., TOEFL11 and Argument Annotated Essays) with regard to these dimensions. Then, they reviewed the existing AES systems in terms of learning methods employed (e.g., neural network models and transfer learning) and the features considered (e.g., length-based features such as essay length and prompt-relevant features).

With the advent of deep learning methods and improvements in NLP architectures, significant progress has been made in this domain for the educational and commercial success of AES systems. Dong, Zhang, and Yang [17] employed RNN and CNN models for the AES task and developed a hierarchical sentence-document model and attention mechanisms. Taghipour and Ng [18] considered several deep neural network models for AES and showed that their LSTM architecture achieves the best performance over the Automated Student Assessment Prize (ASAP) dataset. More recent studies in the AES domain focus on pre-trained language models such as BERT and its variants. Mayfield and Black [19] report negative results on the effectiveness of BERT for enhancing AES performance over traditional NLP techniques and suggest that computationally expensive transformer fine-tuning can be only justified in specific AES problems. On the other hand, several other studies attribute strong performance gains to BERT variants for the AES tasks [20, 21]. Ormerod, Malhotra, and Jafari [22] conducted a detailed numerical study with AES datasets using several fine-tuned pre-trained language models and showed that ensembling over these models provided the best results while requiring fewer parameters than most pre-trained transformer-based models.

3. Methodology

In this section, we provide a description of the AES dataset explored in our experiments and give an overview of BERT, the underlying pre-trained language model used by each of the classification approaches. We then outline the few-shot learning approaches employed in our analysis, namely, SetFit, and PET and describe how we adapt those to our AES problem.

3.1. Automatic essay scoring dataset

The dataset used for this research is a proprietary, multi-labeled dataset of accounting and general finance examination questions. Specifically, each example in the dataset consists of a single, all-lowercase sentence that was taken from a student’s essay answer. The labels for this task are broken into problem instances. In general, a problem instance involves providing a student with a case study on a company’s financial doings.
In each case study, there are several issues with the company’s decisions, and the student is tasked with both identifying these issues, and outlining how they should be corrected. Accordingly, for each problem instance, there are a number of components that a student must address to achieve full marks. This is complicated by the fact that for components sharing the same problem instance, sentences can belong to several different labels. Accordingly, this dataset is multi-labeled, meaning that a sentence is not limited to a single label. The one exception to this is irrelevant sentences, which do not correspond to any problem instance. Sentences labeled as irrelevant will always have a single label. Table 1 details the cardinality of each problem instance and their overall support. Excluding the irrelevant class, there are 6 problem instances and 51 labels (i.e., 51 components) overall.

<table>
<thead>
<tr>
<th>Class</th>
<th>Irrelevant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P-Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Components</td>
<td>1</td>
<td>12</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td>51</td>
</tr>
<tr>
<td>Support</td>
<td>17437</td>
<td>2621</td>
<td>1465</td>
<td>1483</td>
<td>2603</td>
<td>1353</td>
<td>859</td>
<td>10384</td>
</tr>
</tbody>
</table>

Table 1. AES data distribution

To facilitate our analysis, we define three labeling tasks among the hierarchy of data labels. Firstly, we consider the binary classification task of relevancy detection, training models to detect if a sentence should be labeled as “relevant” or “irrelevant”. Secondly, we examine a coarse-grained dataset by considering only the problem instances and not their components. We exclude the “irrelevant” label from this analysis. Through a preliminary analysis, we found that no instances had multiple labels across the problem instances, which is intuitive as different problems typically require non-overlapping answers. Accordingly, we model this coarse-grained dataset using single-label classification. Thirdly, we consider a fine-grained dataset which includes all component labels except for the “irrelevant” label. In this case, problem instance and components are combined to form each fine-grained label. The number of labels considered in this task is therefore 51. As this is a multi-labeled dataset, we use multi-label classification to model the fine-grained dataset.

Table 2 provides some example sentences from the dataset, as well as their problem instance and component(s). Through a preliminary analysis, we found that the relevant problem instances had a mean sentence length of $18.7 \pm 10.0$, while the irrelevant sentences had a mean length of $14.4 \pm 9.5$.

Table 2. AES data sample sentences and the labels.

<table>
<thead>
<tr>
<th>Example</th>
<th>Sentence</th>
<th>Problem Instance</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the contractor had not yet constructed the driveway</td>
<td>4</td>
<td>(c)</td>
</tr>
<tr>
<td>2</td>
<td>the proprietor was not available to verify the driveway at the time of installation, and therefore reasonable assurance was not satisfied</td>
<td>4</td>
<td>(i)</td>
</tr>
<tr>
<td>3</td>
<td>per these results, the company should consult with the bank</td>
<td>6</td>
<td>(a)</td>
</tr>
<tr>
<td>4</td>
<td>the previous calculations show that the company barely satisfies the terms outlined in the bank covenant</td>
<td>6</td>
<td>(g), (h)</td>
</tr>
</tbody>
</table>

3.2. Multi-label classification

In multi-label classification tasks, an instance can be associated with more than one label at the same time. Excluding the irrelevant label, there are 8,263 sentences with a single label, 1,776 with 2 labels, 264 with 3 labels, 77 with 4 labels, and 4 with 5 labels in our dataset. There are three main approaches to multi-label classification in literature. The first approach is one-vs-rest (one-vs-all) classification. It assigns one binary classifier per label and considers each class independently when labeling an example. Each label-wise model detects if the corresponding label is suitable or not for a given example. The second approach is classification via classifier chains. As in one-vs-rest, this method utilizes binary classifiers, but it organizes them into a chain. Each model runs sequentially and adds the previous classifier’s prediction to all available features before training and inference. Therefore, contrary to the previous model, decisions are made dependently on the chain ordering. The third approach is to adopt the multi-output strategy.
This consists of fitting one classifier to predict all of the labels simultaneously. In a neural network, this is achieved by using the sigmoid activation function instead of the softmax function at the last layer, turning a multi-class classification problem into a multi-label classification problem.

3.3. BERT model

Bidirectional Encoder Representations from Transformers (BERT) is a case-insensitive language representation model pre-trained on a large corpus of unlabeled data [23]. The pre-trained model can be fine-tuned using labeled data on a wide variety of NLP tasks in a process known as transfer learning. The BERT model is pre-trained with MLM and next sentence prediction (NSP) objectives [23]. Specifically, in MLM, the unlabeled corpus of text data is tokenized, and some of the tokens are masked at random. The model is then trained to predict the masked token based on the context. Whereas NSP involves randomly selecting pairs of sentences from the dataset and predicting whether the second sentence follows the first. We employ the BERT-large checkpoint, which has 340 million parameters, for each of the classification approaches used in this work to ensure a fair comparison of the different modeling approaches [23]. However, we acknowledge that the relative performance of each algorithm may change depending on the adopted pre-trained language model checkpoint. As a baseline comparison to the few-shot learning approaches used in this work, we use BERT for sequence classification, often referred to as fine-tuning.

3.4. Pattern-exploiting training

PET involves reformulating input examples into a cloze-style phrase and leveraging the MLM capabilities of pre-trained language models to improve few-shot performance [2, 3]. This reformulation process consists of two mappings, which are referred to as patterns and verbalizers [2]. In this research, we only consider PET in a single-label classification context. For such problems, a training example consists of a sentence \( x \) and a label \( \ell \). A pattern \( p(x) \) is a mapping which takes \( x \) as an input and outputs a phrase containing a masked token, known as a cloze-style phrase. A verbalizer \( v(\ell) \) is a mapping which maps a label \( \ell \) to a token from the vocabulary. The likelihood that the MLM predicts \( v(\ell) \) is then used to generate a probability that the correct label is \( \ell \). A key advantage of PET is its ability to insert context into a problem through reformulation. In general, context is learned through fine-tuning on a large amount of training data. In contrast, PET can provide the context through the design of a pattern.

![Figure 2. PET example.](image)

To explain the mechanism behind the PET algorithm, we use the example in Figure 2. Here, the objective is to take an input sentence \( x \) which states the total cost of a project, and determine if it exceeds 5 million dollars. Assume that the two labels are 1, indicating that the cost did exceed 5 million dollars, and 0 indicating that it did not. With fine tuning, this task is accomplished by labeling a large number of statements \( x \) and fine-tuning the model on these labeled examples. Notably, while this task is simple for someone with an understanding of natural language, it might not be possible for a traditional sequence classifier to
determine the task without fine-tuning. PET attempts to address this by taking the input statement \( x \) and reformulating it to the cloze-style phrase

\[
p(x) = \text{did the total cost exceed 5 million dollars?}, \ x.
\]

The verbalizer is then

\[
v(0) = \text{no} \\
v(1) = \text{yes}
\]

Rather than fine-tuning the model on a large amount of labeled data, we instead ask a pre-trained language model to predict the probability that the masked token should be “yes” and “no”. Whichever is more likely tells us the label. In the given example, the sentence \( x \) should be labeled 0, indicating that the cost is less than 2 million dollars. The MLM generates a probability that the masked token should be “no” of 90%, and the example \( x \) is correctly labeled with a 0. In practice, labeled training data is used to fine-tune the MLM.

In a few-shot setting, the performance of a given model can vary considerably by the dataset, pattern designs, and random seeding [2–4]. Accordingly, it is often advised that three patterns should be designed, and that for each pattern three MLM models should be trained using different random seeds [2, 24]. However, this gives nine models in total which are necessary to form predictions. In this case, prediction time scales linearly with the number of models, incurring a massive overhead to many alternative classification algorithms that use pre-trained language models. To address this, PET can leverage access to unlabeled data to condense the knowledge of these nine models into a single model through a process known as knowledge distillation [2, 3]. This process assumes that unlabeled data is available in abundance, but this is often found to be the case in practice [2, 3].

Given access to unlabeled data, each MLM, \( m \in M \), is used to predict scores \( s_m(\ell|x) \) for each label. These scores are combined according to the accuracy of the MLM on the training set before training is done, thereby giving higher weight to patterns which better capture the context of the problem:

\[
s(\ell|x) = \sum_{m \in M} w(m)s_m(\ell|x)
\]

These scores are then converted to probabilities using softmax, and a sequence classifier is trained on the set of training data and this set of softly-labeled data (formerly the unlabeled data).

In the example in Figure 2, one could reasonably come up with a verbalizer, likely the same as the one in the example, very quickly. However, in the AES problem, even with access to a domain expert we came across tremendous difficulties designing a verbalizer for each problem instance, complicated both by the fact that some problem instances cover similar topics, and that reducing a multi-step problem to a single, appropriate word is difficult. This is frequently discussed as a major limitation of the PET algorithm [4, 13]. However, it was found that verbalizers could be automatically found using what is known as the PET with Automatic Labels (PETAL) algorithm [25]. The PETAL algorithm evaluates the training data and unlabeled data to identify verbalizers for each of the labels [25]. In our experiments, we use PETAL to generate verbalizers that map each label to a set of 10 words, following the example set in [25].

3.5. SetFit

While PET offers the ability to introduce context to a problem, it adds overhead in terms of computing power and often requires domain expertise for designing patterns and verbalizers [4, 13]. SetFit attempts to remedy these issues by creating a few-shot learning model that, to the end-user, works very similarly to fine-tuning in that the domain expertise is not needed, external inputs like patterns and verbalizers need not be designed, and only one model needs to be trained.

As in any few-shot learning approach, the objective of SetFit is to leverage the scarce amount of labeled data available. To do this, a contrastive approach for training is used [4]. Given two training instances \( x_1 \) and \( x_2 \) with the same label, we can construct a positive triple \( T_p = (x_1, x_2, 1) \). Similarly, if these instances have differing labels, we can construct a negative triplet \( T_n = (x_1, x_2, 0) \). In the SetFit algorithm, for each class label, \( R \) positive and negative triplets are generated by pairing together different examples from the dataset. In doing so, the dataset is enlarged considerably, and the triplets can be used to fine-tune a sentence transformer (ST): pre-trained sentence embeddings [4]. Once the sentence transformer is fine-tuned, it is
used to encode the training data, and a logistic regression model is trained on the encoded data [4]. At inference time, the same procedure is followed: an unknown example $x$ is encoded via the ST, and the logistic regression model predicts the class.

4. Numerical Study

In this section, we provide the results of our numerical study. First, we describe the experimental setup for our empirical analysis. Then, we discuss the results from the binary classification task of classifying instances as "relevant" and "irrelevant". Next, we provide detailed results on the effectiveness of few-shot learning methods on the coarse-grained single-label classification task. Lastly, we discuss the effectiveness of few-shot learning for multi-label classification with the fine-grained dataset.

4.1. Experimental setup

We provide a detailed description of our experimental setup to ensure the reproducibility of our results and enable future research in the field. Below, we discuss hyperparameter tuning, considered PET patterns, and data pre-processing steps.

4.1.1. Hyperparameter selection

Each approach uses the bert-large-uncased checkpoint as the pre-trained language model. Following Schick and Schütze [2], we use a learning rate of $10^{-5}$ and a batch size of 16 for every experiment. As the sentence data for this task is very short, we use a maximum sequence length of 128 in all models.

- **Fine-tuning:** We train each sequence classifier for 1000 steps in both the binary classification and the single-label classification experiments. In the fine-grained experiment, we found that training for 1000 steps was insufficient. Accordingly, in the multi-label classification experiment, we use 10,000 training steps.

- **PET:** Following the advice in Helmeczi, Cevik, and Yıldırım [24], we train each MLM for 1000 steps, and train the final sequence classifier on the unlabeled data with 5000 training steps. We choose our remaining hyperparameters according to those advised in Schick and Schütze [2] and Schick and Schütze [3]. We use a temperature value of 2 in the softmax function for combining the soft-labeled data. We use an adam epsilon of $10^{-8}$ and set the maximum gradient norm to 1.

- **SetFit:** For each experiment, we train for 3 epochs and, following Tunstall et al. [4], we use $R$ value of 20 for contrastive learning. Unlike fine-tuning, we did not find that training longer by increasing the number of epochs in a multi-label setting increased the performance of SetFit substantially. We evaluated three types of multi-label strategies in the literature. We found that the one-versus-all strategy gets the best result for few-shot multi-label training settings. Hence, all results of the few-shot setting are based on one-versus-all strategies.

4.1.2. PET patterns

For each task, we use a set of three patterns and train three MLM models for each with different random seeds for a total of nine models to generate soft labels. In the binary classification task, we use three custom made patterns for the PET algorithm

1. Is "x" a solution?
2. "x" fully addresses the question.
3. "x"?

with a single custom made verbalizer

$v$(Irrelevant) = No
$v$(Relevant) = Yes

For the single-label classification and multi-label classification tasks, we use the same custom made patterns

1. On the topic of ____ the student wrote “x”.

7
and use PETAL to generate a verbalizer for each pattern, which maps each label to a set of 10 words. As PETAL takes the training data as input, the verbalizers for each training dataset are potentially unique [25].

4.1.3. Data preparation for few-shot setting

Few-shot learning involves training a model with access to limited data. Given that our task is automatic essay scoring, we assume that each label will be represented in our dataset. For the binary classification task, we perform a random sampling of the dataset, to gather training sets of sizes 25, 50, 100, 200, and 400. For the multi-label dataset, we ensure each label has support by drawing a certain number of shots per label. We consider 1 through 10 shots per label with the intention of keeping the number of training instances small to evaluate the few-shot performance of the models.

For the single-label classification problem, we consider an additional sampling strategy. As with the fine-grained experiment, we consider datasets constructed with an equal number of shots per label, considering 3, 5, 10, 20, and 50 shots per label. Instead, in practice, constructing such a training set is often not practical. Instead, the distribution of labels will generally reflect random sampling: labels that occur more naturally, in general, will make up a higher proportion of the training data. Accordingly, we construct a coarse-grained dataset with 3 shots per label (for a total of 18 labeled examples) with the rest of the data being randomly sampled. While some few-shot tasks allow for some labels to go unrepresented in the training data entirely, for an AES task with easy access to a prompt and resources like ChatGPT available, we assume that some example sentences could be produced for these labels to ensure that at least 3 labeled examples are available. As with the relevancy detection task, we consider datasets of size 25, 50, 100, 200, and 400 labeled examples.

For each problem, we randomly sample 2,000 data instances to serve as our test data, and for the binary and single-label classification tasks, we set aside 5,000 unlabeled examples for knowledge distillation with PET. As few-shot learning results can often vary considerably between datasets, we repeat each experiment three times, constructing three unique training, unlabeled, and test datasets, and report the average results.

In single-label classification, the number of training instances corresponds to the number of labels times the number of shots per label. In multi-label classification however, as some examples will have more than one label, the number of training instances is less than the number of labels times the number of shots per label. Accordingly, we provide an approximate number of training instances in our results by taking the average number of training instances for each training set size.

It should be noted that SetFit was designed with extremely small datasets in mind, with access to only about 50 or fewer labeled examples [4]. Such sizes are generally considered few-shot [26]. In our experiments, we consider datasets of sizes up to 400, extending a bit beyond the few-shot learning domain. Since the complete dataset that we derive our few-shot learning dataset from contains more than 20,000 labeled examples, while labeling 400 examples may not always be feasible in a truly few-shot domain, evaluating the performance of few-shot learning models on such datasets offers valuable insights.

4.2. Relevancy detection results

To begin our experiments, we consider the task of determining whether a given sentence in an essay is relevant or irrelevant to the considered problem instances. Figure 3 reveals that the PET model tends to perform better than the other models considered, offering as much as a 5% performance improvement over fine-tuning. Notably, the performance of SetFit does not always exceed that of fine-tuning for this task. These results suggest that for the binary classification task, SetFit does not offer any benefit to fine-tuning alone, beyond reducing the run-time. It is worth noting that, unlike SetFit and fine-tuning, the PET algorithm requires access to an abundance of unlabeled data in order to train a final predictor. However, as noted earlier, this data should be generally available for the AES task.

One major benefit to the SetFit algorithm is its ability to quickly train models with access to limited data. In our experiments, we found that for smaller dataset sizes, SetFit was the quickest to train. As the number of labeled instances is increased, however, the generation of additional contrastive pairs increases the training time of SetFit beyond fine-tuning.
4.3. Coarse-grained single-label classification results

We next consider a single-label classification task involving problem instance prediction using our coarse-grained dataset. While determining the problem instance alone is not sufficient to grade a student’s essay, it does offer two clear advantages. Firstly, determining a sentence’s problem instance allows for a more focused multi-labeling model to be fine-tuned afterwards on sentences pertaining to that problem instance alone. As the maximum number of components in a problem instance is 12, compared to 51 in the entire dataset, the resulting model might have an easier job labeling examples. Secondly, as the different problem instances can vary considerably in their area of focus, it is not necessarily the case that a human marker can mark all of the problem instances. By labeling sentences according to their problem instance proactively, we can ensure that human markers only consider answers relevant to the questions that they can mark, which can reduce the workload on markers considerably.

Figure 4 shows the results for problem instance detection using an equal number of shots per label, i.e., balanced training data. These results demonstrate that PET once again is the best performing model, offering a more than 10% performance boost over fine-tuning with access to only 18 labeled examples (i.e., 3 shots/label). Unlike in the case of binary classification, SetFit is generally superior to fine-tuning in this case only dropping below the accuracy of fine-tuning for 100 examples (i.e., 10 shots/label). As with binary classification, we note that SetFit trains extremely fast for smaller training set sizes, but the runtime increases as we add more labeled examples.

Figure 5, which shows the results for imbalanced randomly sampled training sets, reveals even more promising results for PET, which gives a 17% accuracy boost over fine-tuning alone with access to only 25
labeled examples. In this case, SetFit generally performs as good or slightly better than fine-tuning alone but, as with the results using an equal number of shots per label, the main advantage to SetFit is its training speed when used on small datasets. Both sampling strategies reveal PET as the superior algorithm for problem instance detection over coarse grained dataset.

4.4. Fine-grained multi-label classification results

While in theory PET could be used for multi-label classification by applying a one-vs-rest approach, accomplishing this on a 51-label dataset would require training 451 MLM models and a final sequence classifier. As this process would be very time-consuming and likely infeasible in practice, we exclude it from our analysis. The results in Figure 6 show that SetFit performs worse than fine-tuning alone in a fine-grained multi-label context. However, we did find that for the 51 label case (i.e., 1 shot/label) where the performance of SetFit and fine-tuning are similar, SetFit was able to train in about one tenth of the time that fine-tuning took. Unfortunately, neither model performs well in a few shot setting, with neither model passing 50% in terms of f1-scores until given access to about 200 labeled instances.

5. Conclusion and Future Research

In this study, we investigate the viability of few-shot learning approaches for automatic essay scoring. Our results show that few-shot learning methods offer a significant advantage over traditional fine-tuning both
for relevancy detection, and in problem instance detection (i.e., coarse-grained labeling). In all settings, we find that SetFit offers the fastest training speed when data is extremely limited. However, as the number of labeled instances increases, the SetFit algorithm begins to fall behind fine-tuning in terms of training speed. Additionally, SetFit fails to outperform fine-tuning at any training set size in a fine-grained multi-label context, while PET is infeasible for multi-labeling on such a large dataset.

There are numerous opportunities for future research stemming from this work. Firstly, while this study considers both PET and SetFit in several settings, we find that it is challenging to run PET in a multi-label setting. Accordingly, future studies might consider alternative few-shot multi-labeling approaches to provide insights into how these algorithms respond to high-cardinality multi-labeled datasets. Secondly, the exploration into multi-labeled few-shot learning for the AES task in this research focuses on a dataset with 51 labels. With such high cardinality, this evaluation offers little insights into the performance of few-shot learning for AES on a single question, where the number of labels would be significantly reduced. Accordingly, future research into few-shot learning approaches for AES might consider multi-label classification for smaller problems. Thirdly, aside from the relevancy detection task, the verbalizers for the PET algorithm are generated using PETAL for this task. The PETAL algorithm is often found to perform worse than manually creating verbalizers, as manual labeling offers the ability to provide additional domain knowledge not contained within the training and unlabeled datasets [25]. While our results show that PET is generally able to outperform fine-tuning alone, future research might consider the impact of hand-crafted verbalizers on few-shot performance for the AES task.

Acknowledgements

This research was enabled in part by support provided by the Digital Research Alliance of Canada (alliancecan.ca). We would like to thank the reviewers for their valuable insights and suggestions. We would also like to thanks blees.ai for providing the dataset and offering their time and expertise in interpreting the dataset.

References