QATAR UNIVERSITY

COLLEGE OF ENGINEERING

RELEVANCE SCORING OF ARABIC AND ENGLISH WRITTEN ESSAYS WITH DENSE RETRIEVAL

BY

SALAM M. ALBATARNI

A Thesis Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Masters of Science in Computing

June 2025

COMMITTEE PAGE

The members of the Committee approve the Thesis of
Salam M. Albatarni defended on 12/05/2025.

Dr. Tamer Elsayed
Thesis Supervisor

Prof. Rehab Duwairi
Committee Member

Prof. Abdelaziz Bouras
Committee Member

Prof. Mohamed Haouari
Committee Member

Prof. Vincent Ng
Committee Member

Approved:
~~Dean, College of Engineering~~

ABSTRACT

Albatarni, Salam, M., Masters : June : 2025, Masters of Science in Computing
Title: Relevance Scoring of Arabic and English Written Essays
with Dense Retrieval
Supervisor of Thesis: Dr. Tamer Elsayed.
Automated Essay Scoring automates the grading process of essays, providing a great advantage for improving the writing proficiency of students. While holistic essay scoring research is prevalent, a noticeable gap exists in scoring essays for specific quality traits. In this thesis, we focus on the relevance trait, which measures the ability of the student to stay on-topic throughout the entire essay. We propose a novel approach for graded relevance scoring of written essays that employs dense encoders. Dense representations of essays at different relevance levels then form clusters in the embeddings space, such that their centroids are potentially separate enough to effectively represent their relevance levels. We hence use the simple 1-Nearest-Neighbor classification over those centroids to determine the relevance level of an unseen essay. We evaluate our approach in both task-specific (training and testing on the same task) and cross-task (testing on unseen tasks) scenarios using English (ASAP) and Arabic (in-house) datasets. For English, our method achieves state-of-the-art performance in the task-specific setting and matches baseline performance in the cross-task setting, while a few-shot analysis shows it reduces labeling costs with only a 9% drop in effectiveness. For Arabic, our approach outperforms the baselines with 5 and 2 points in task-specific and cross-task settings respectively.

DEDICATION

To the Syrians who were forced to scatter across the world, to those imprisoned for a lifetime, and to those who fought tirelessly for a livelihood and finally found their freedom.
And to my Palestinian brothers and sisters, against whom the forces of the world conspire to break them, yet they stand unyielding, determined to free our land.

ACKNOWLEDGMENTS

First and foremost, praise be to Allah, who has guided me in working on this thesis. I would like to express my deepest gratitude to my family for their patience and encouragement. Their support has been a constant source of strength and motivation throughout this journey.

I am also sincerely grateful to my supervisor, Dr. Tamer Elsayed, for his invaluable guidance, patience, and encouragement throughout this research. Additionally, I appreciate my friends and colleagues, Sohaila Eltanbouly and May Bashhendy, for their continuous support and collaboration, which made this journey enjoyable.

This thesis was made possible by the support of the Qatar National Research Fund (a member of Qatar Foundation) through the GSRA grant# GSRA10 L-2-0521-23037 and the NPRP grant# NPRP14S-0402-210127. Additionally, this work was supported by Qatar university student grant QUST-1-CENG-2025-357. The statements made herein are solely the responsibility of the author.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Automated Essay Scoring (AES) has been a prominent field of research for over five decades [1]. AES aims to automatically assign a quality score to written essays. It comes in handy for teachers, alleviating the burden of correcting numerous essays and allowing them to concentrate on more crucial responsibilities. Furthermore, AES has the advantage of providing quick and consistent feedback to students, improving the learning process. AES research has primarily focused on holistic scoring [2], which provides a single score reflecting the overall quality of the essay. However, in terms of practicality and effectiveness, a single score falls short in guiding students on how to enhance their skills. Trait-based AES, shown in figure 1.1, fills this gap by individually scoring the *traits* quality, e.g., organization, development, and relevance [3]. Trait scoring enables students to gain insights into specific areas of improvement, empowering them to understand their weaknesses and enhance their writing proficiency.

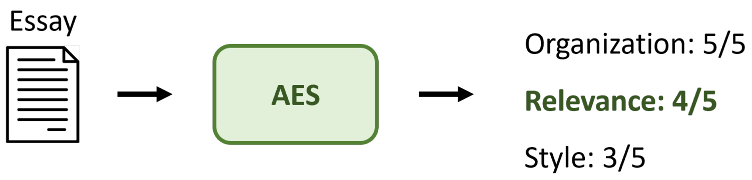


Figure 1.1. Trait-based AES system.

In this work, we focus on scoring the *relevance* (or so-called *prompt adherence*) trait. The relevance trait, in particular, evaluates the extent to which the essay aligns with the given *task-prompt*.[1] This trait is crucial as it gauges the student's ability to stay on topic, i.e., maintaining a clear and direct connection to the main subject throughout the writing. Several studies used the essence of prompt adherence as a means to aid in holistic scoring [4]; however, there is a lack of tailored scoring approaches for the relevance trait.

AES systems can be categorized into two types based on the training and inference settings as shown in figure 1.2. *Task-specific* AES focuses on a single writing task, where the system is trained exclusively on essays written for *one* specific task, then, during inference, it grades unseen essays written for the *same task*.

This setup enables the AES system to learn the distinctive characteristics associated with a specific task, allowing for a more precise assessment of the essays written in response to that task. This type is predominant in the literature for both holistic and

[1]The task-prompt refers to the specific instructions or guidelines provided to students to guide their essay writing on a particular *topic*.

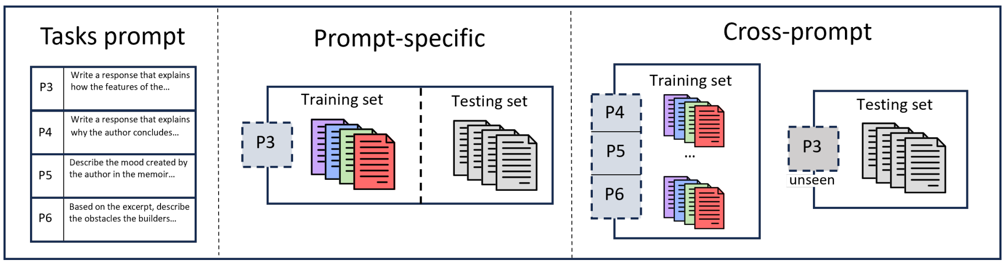

Tasks prompt
P3
Write a response that explains how the features of the...
P4
Write a response that explains why the author concludes...
P5
Describe the mood created by the author in the memoir...
P6
Based on the excerpt, describe the obstacles the builders...
Prompt-specific
Training set
Testing set
P3
Cross-prompt
Training set
P4
P5
P6
...
Testing set
P3
unseen

Figure 1.2. Task-specific scores vs cross-task scoring

trait-based AES. The early research on task-specific AES used feature-based learning approaches [5], [6]. Then, different neural-based approaches have been proposed [3], [7], [8]. Subsequently, pre-trained language models have become dominant for developing AES systems [9]–[12]. Most of those works focused on holistic scoring, and less attention has been given to scoring individual traits.

Meanwhile, *cross-task* (also known as *cross-prompt*) AES systems are trained on essays written for multiple *source* writing tasks to grade essays written for *unseen target* tasks [12]. In this setting, the AES system leverages insights gained from various writing tasks to score essays from unseen tasks. It is commonly observed in real-world scenarios that limited data is available for target tasks [13], which emphasizes the necessity for developing generalizable cross-task AES systems for grading essays for unseen tasks. Multiple research studies have proposed cross-task holistic scoring systems, e.g., [13]–[15]. More recently, other approaches for cross-task AES systems have been proposed to score individual traits of the essay along with the holistic score, e.g., [16]–[18]. Nevertheless, all traits are handled the same, without employing a distinct approach for any specific trait, overlooking the fact that each trait focuses on a different dimension of the writing quality of the essay.

While the aforementioned research has significantly advanced AES for the English language, the Arabic language has received considerably less attention in this domain. In contrast to the extensive body of work on English AES, scientific research dedicated to Arabic AES remains minimal, leaving a notable gap in the field. Despite this, there has been little work on Arabic task-specific AES, most of them utilizing neural networks [19], [20]. Some studies extracted hand-crafted features, such as the number of spelling mistakes [21], [22]. Few studies addressed traits scoring including structure [23], and very recently the relevance trait [24]. While these studies represent a commendable initiative in addressing the problem, some of them fall short in adhering to well-established evaluation metrics within the field. Additionally, they often fail to provide critical details regarding the datasets used, as well as the methodologies employed for training and testing [23], [24]. Furthermore, to date, none of these studies explored Arabic cross-task scoring, whether in the context of holistic or trait-based scoring.

To this end, in this work, we focus on scoring the relevance trait for both *task-specific* and *cross-task* scenarios across two languages, namely, English and Arabic. We propose a *novel* approach for *graded relevance* scoring (i.e., scoring into one of multiple relevance grades or levels) of written essays. It employs dense encoders to represent training essays in the embedding space. We then hypothesize that the dense representations of essays having the same relevance level form a cluster in that space such that the centroids of the clusters of different relevance levels will be separate enough to effectively represent their respective relevance levels. We hence use a simple $1NN$ classification model over those centroids to determine the relevance level of an unseen essay.

We leverage multiple dense encoders, which have demonstrated effectiveness in retrieval and sentence representation tasks, and apply them to score the relevance trait.

In the context of graded relevance scoring of written essays, we address the following research questions, for both languages, Arabic and English:

- **RQ1**: Can the *pre-trained* encoders be effectively leveraged for the task? (**Out-of-the-box scenario**)
- **RQ2**: What is the impact of *fine-tuning* the encoders on the performance? (**Fine-

tuning scenario)

- **RQ3**: In case *no* training essays are available, can we effectively leverage previously-labeled essays from other tasks? (**Cross-task scenario**)

Given that the English data set is much larger, we also address an additional question specific to English essays:

- In case only *few* training essays are available for each relevance level, how would that affect the performance? (**Few-shot learning scenario**)

Our study yields promising results when scoring the relevance trait using pre-trained encoders; however, fine-tuning consistently outperforms the baselines for both languages. Furthermore, we propose an extension to our approach for the cross-task setup. We transform the approach to *task-independent* by "excluding" the information of the task-prompt from the essays. This simple trick resulted in a performance boost, achieving an on par performance with a baseline model for that scenario.

In summary, our main contribution are:

1. We propose a novel approach that employs dense retrieval for the general task of graded relevance scoring.
2. For English relevance trait AES:
   (a) Our approach establishes a new SOTA performance on scoring the relevance of English essays in the task-specific setup.
   (b) We analyze the performance of our approach in a more practical few-shot scenario, showing a significant saving of labeling cost while sacrificing only 9% of the effectiveness.
   (c) We propose a simple but effective cross-task extension of our approach that made it on par with the SOTA performance.
3. For Arabic relevance trait AES:
   (a) Our proposed approach, pre-trained and fine-tuned, outperformed the Arabic baselines, demonstrating its robustness.
   (b) In the cross-task scenario, our simple approach outperforms all baselines across different encoders in different setups.

The rest of the thesis is organized as follows. Chapter 2 discusses the work related to relevance trait scoring for task-specific and cross-task setups for both English and Arabic languages. A detailed description of our approach is provided in Chapter 3. Chapter 4 presents the experimental setup and answers our research questions for English relevance scoring while Chapter 5. Finally, we conclude in Section 6.

CHAPTER 2: RELATED WORK

In this section, we offer a comprehensive review of existing methods for task-specific and cross-task techniques for relevance scoring. We address the limitations of prior work and highlight the unique contributions and distinctions of our approach.

2.1. English AES

2.1.1. Task-specific Scoring

When it comes to AES, task-specific holistic scoring is predominantly emphasized, with less attention directed towards trait scoring. The early research concerning the scoring of the relevance trait focused on feature-based approaches. Persing and Ng [5] pioneered the focused exploration of the relevance trait. They used a linear SVM regression model with a rich set of lexical and knowledge-based features to measure the relevance between the essays and the task-prompt. Mathias and Bhattacharyya [6] used a common feature set with a Random Forest (RF) classifier for predicting the holistic and traits scores.

Others employed traditional retrieval approaches, such as TF-IDF and pseudo-relevance feedback (PRF), to measure the similarity between the essay and the task-prompt. Cummins, Yannakoudakis, and Briscoe [25] expanded the task prompt, and the relevance score was computed as the cosine similarity between the TF-IDF representations of the essay and the expanded prompt. The expansion terms were selected based on the closest words to the prompt vector, which was constructed using random-indexing, CBoW, skip-gram, and PRF. Similarly, [26] used TF-IDF, CBoW, and skip-thoughts models to measure the similarity between the prompt and each sentence in the essay. Chen and Li [4] used the relevance of the essay to the prompt as a feature for holistic scoring. The essay and prompt representations were acquired through an attention-based RNN, with the relevance score being computed via element-wise multiplication between the essay and prompt vectors.

In light of the limited research on trait scoring, the objective of Mathias and Bhattacharyya [3] was to leverage various models originally intended for holistic scoring to score multiple essay traits, including relevance. Three approaches were used: a feature-based model with RF algorithm [6], a string kernel-base approach [27], and an attention-based neural model [7]. Kumar, Mathias, Saha, *et al.* [10] used a multi-task neural model to predict multiple writing trait scores in parallel (auxiliary tasks), and these scores were used by the network to predict the holistic score (primary task). They also tried to set one trait as a primary task and other traits along with the holistic score as auxiliary tasks. Most recently, Do, Kim, and Lee [28] employed a large language model, specifically T5, fine-tuning it to score all traits along with the holistic score. They prompted the model with the essay, prompt text, and the holistic and trait scores. Although their model proved to be extremely effective, setting new state-of-the-art results for nearly all traits, their training setup differs somewhat from traditional task-specific scoring. Instead of training a separate model for each task, they trained one model on all available tasks and then tested it on all tasks as well. While this approach efficiently avoids the need to train multiple models, it does not fully align with the task-specific scoring definition, where the model should be trained exclusively on the target task [2], [29]. In a follow-up study, the same author employed Reinforcement Learning (RL) to design an evaluation method based on the metrics commonly used in AES systems. While this approach slightly improved upon the results from fine-tuning T5, it retains the same limitation: a single model is used for all tasks, rather than one model per task [30].

The related work reveals two major weaknesses in relevance scoring. First, although there are some approaches that have targeted the relevance trait, the majority of those approaches rely on feature-based and traditional retrieval methods [5], [26]. Second, none of the recent approaches focused on the relevance trait in specific; rather, they offered a generalized model for all traits, ignoring the fact that each trait pertains to different aspects within the essay [10], [28], [30]. In contrast, our method eliminates the need for feature engineering by leveraging recent advances in dense information retrieval, offering a dedicated approach for modeling the relevance trait in AES.

2.1.2. Cross-task Scoring

Cross-task AES aims to train a model using labeled essays from one or more source tasks and then apply the model to score essays from a target task. Similar to task-specific research, predominant studies on cross-task evaluation have focused on holistic scoring [13], [14], [31]. The cross-task trait-based AES was introduced by Ridley, He, Dai, *et al.* [16], where all of the efforts previously focused solely on holistic scoring. Their approach is an extension of the work of Ridley, He, Dai, *et al.* [31], which utilized part-of-speech embedding with a convolution network to generate the essay's representation. The architecture was modified by introducing shared low-level layers to learn common representation across tasks and high-level layers to capture task-specific information. ProTACT model [17] also employed the idea of hierarchical representation, employing low-level layers for information sharing across traits and top layers for trait-specific information. Moreover, the task-prompt was utilized to acquire prompt-aware representation by applying essay-prompt attention. Chen and Li [18] proposed PMAES, a framework designed to improve cross-task representation through a prompt-mapping contrastive learning strategy. This approach involved projecting source task essays onto target tasks, and generating mapping representations specific to the target task. The objective was to minimize the distance between these mapping pairs, thereby aligning source and target tasks to achieve greater consistency in their representations. Li and Ng [32] explored a variety of text features to develop a purely feature-based model that could compete with more complex models. They achieved competitive results with a simple network, reaching state-of-the-art performance on the ASAP dataset for the relevance trait. Most recently, Do, Park, Ryu, *et al.* [33] used a T5-based pre-trained model to obtain a grammar-corrected version of the input essay. The model learns to compare the original and grammar-corrected essays and infer the output score.

Trait cross-task AES has received less attention compared to task-specific AES. Similar to the task-specific approaches, all of the existing solutions for cross-task provided a common framework for all the traits. Our cross-task approach is *tailored specifically for the relevance trait*. This is done by generating task-independent representations, which effectively map essays with similar relevance levels across different tasks closer together. One of the unique aspects of our approach is its adaptability for both task-specific and cross-task scenarios, requiring only minor modifications.

2.2. Arabic AES

Arabic AES faces a significant challenge that has hindered progress in the field: the lack of publicly available annotated essay datasets. Unlike English AES, which benefits from a wealth of resources, Arabic AES research has been considerably slower due to this limitation. Most studies in this domain rely on in-house datasets, which, while valuable for individual research efforts, are not accessible to the broader community. This lack of standardization prevents meaningful comparisons between models and limits advancements that could arise from collaborative and reproducible research.

With that being said, the work on Arabic AES can be broadly categorized into traditional feature-based approaches, neural network-based methods, and language model-based methods.

Traditional approaches have predominantly relied on rule-based techniques and feature engineering. For instance, [34] utilized the Longest Common Subsequence (LCS) algorithm combined with Arabic WordNet to evaluate short Arabic answers, achieving high accuracy in scoring. Similarly, [21] extracted features from lexical, syntactic, and semantic levels, combining them to produce a final essay score, demonstrating the effectiveness of multi-level linguistic analysis. Other studies, such as [35], explored similarity measures like Euclidean, Jaccard, and Cosine distances alongside Arabic WordNet for semantic analysis, with Cosine similarity yielding the lowest error. Hybrid stemming techniques, as proposed by [36] and [37], combined Extended Light Stemmer, ISRI Stemmer, and Arabic WordNet to reduce words to their root forms, significantly improving scoring precision. Additionally, [23] employed Support Vector Regression (SVR) with morphological, syntactic, semantic, and discourse features to evaluate essays across multiple criteria, achieving a 96% accuracy rate. Lastly, [22] introduced a system that uses Latent Semantic Analysis (LSA) and Rhetorical Structure Theory (RST), achieving a 90% accuracy rate and a 0.756 correlation with human scoring. These traditional methods, while effective, often require extensive feature engineering and struggle with deeper semantic understanding.

In contrast, neural network and language model-based approaches have begun to address these limitations by leveraging advanced machine learning techniques. For example, optimization algorithms like Particle Swarm Optimization and the e-Jaya algorithm was used to train neural networks, demonstrating the potential of optimization techniques in improving model performance [19], [20]. More recently, transformer-based language models have gained traction in Arabic AES. Machhout and Zribi [24] proposed an enhanced AraBERT-based approach, augmented with handcrafted features, to score the relevance of Arabic essays to their prompts, achieving a remarkable 0.88 correlation with human scores.

While the presented methods are valuable early efforts in Arabic AES, there remains a significant gap to be addressed. First, nearly all previous approaches rely on feature engineering [19], [23], [34]. Additionally, much of the existing work focuses solely on holistic scoring [19], [20], [34]. Even studies targeting the relevance trait apply simple fine-tuning of AraBERT without accounting for the specific characteristics of relevance scoring [24]. Finally, to our knowledge, no prior work has explored cross-task setups in Arabic AES, and certainly none for the relevance trait specifically.

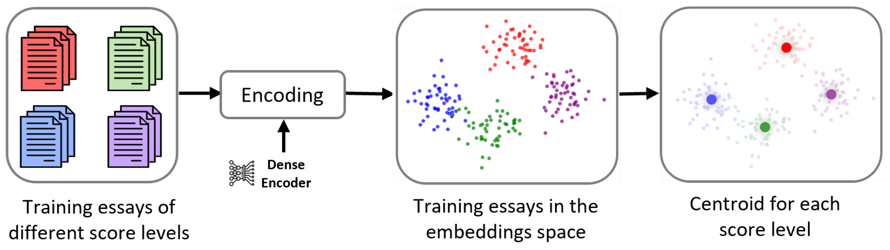
We address these gaps by proposing a method that leverages recent advancements in dense retrieval, eliminating the need for feature engineering. Moreover, to the best of our knowledge, we are the first to propose a method for cross-task scoring of the relevance trait in Arabic.

CHAPTER 3: METHODOLOGY

3.1. Proposed Approach

In this section, we present and discuss in detail our proposed approach for graded relevance scoring of essays in different scenarios. We address the task-specific, fine-tuning, cross-task scenarios in the following sections.

3.1.1. Graded Relevance Scoring of Essays with Dense Encoders



Training essays of different score levels
Encoding
Dense Encoder
Training essays in the embeddings space
Centroid for each score level

Figure 3.1. Training phase of our proposed approach for graded relevance.

Dense retrieval models are generally encoders that are trained to generate dense vector representations of documents (and queries) so that texts that are topically similar are close in the embedding space, while those that are topically distant are further apart in that space. This can then be employed to find documents that are relevant to a given query, in a typical retrieval scenario, by computing the similarity between the dense representations of the query on one side and documents on the other side.

Inspired by this concept, our approach builds on this idea to score the relevance trait of students' written essays given a task-prompt, but from a different angle. While the obvious approach is to use the prompt as a query and the test essays as documents in the above setup, several challenges arise. Notably, prompts often differ in length from essays, making the mapping of similarity scores to relevance levels harder. Furthermore, this approach may not capture the diverse writing proficiency levels exhibited in essays, particularly those written by school students. This variability adds complexity to the problem.

Our approach relies heavily on *two* main components: having examples of labeled (i.e., manually-graded or scored) essays from *each* relevance level, and utilizing a robust dense encoder. We hypothesize that the encoder can effectively encode essays in a manner that positions those from the same relevance level in close proximity while keeping essays from different relevance levels distant. This yields a (somewhat) separate cluster of essays (in the embedding space) for each different relevance level. Therefore, *centroids* of those clusters can serve as reference points (or good representatives) of their corresponding relevance levels.

This constitutes the "training" phase of our approach. During inference (when we get unseen essays to score), we simply classify essays by assigning them the relevance level corresponding to the *nearest* centroid. In other words, our approach employs a *1-Nearest-Neighbor* (*1NN*) algorithm over the centroids corresponding to the different relevance levels.

This innovative setup reframes the problem; it treats the test essay (i.e., the one to be scored) as a query and the centroids as documents. Our goal is to identify the most "relevant" document (centroid) for the given query (test essay), effectively assigning a relevance score based on this proximity in the embedding space. This approach also transforms the problem into a multi-class classification scenario, where the classes represent the various relevance levels. It is worth mentioning that the model described in this section is a *task-specific* model, as it is trained on essays labeled for a given task-prompt. Figure 3.1 illustrates the training phase of our approach.

Formally, we are given a task-prompt p and a corresponding training set of essays S_p. Each training essay $s \in S_p$ is assigned a relevance level $R(s)$. We denote the set of all training essays that are assigned the relevance level i as S_p^i, where $S_p^i = \{s \in S_p | R(s) = i\}$. We first encode each training essay s using the dense encoder D to get its dense vector representation $\vec{e_s}$, where $\vec{e_s} = D(s)$. Next, for each relevance level i, we compute its corresponding centroid $\vec{C_i}$ as the mean of the essay vectors of that level, as follows.

$$\vec{C_i} = \frac{1}{|S_p^i|} \sum_{s \in S_p^i} \vec{e_s} \tag{3.1}$$

where $|S_p^i|$ is the number of training essays that are assigned the relevance level i.

Subsequently, for a given test essay t, we estimate its relevance level $R(t)$ based on its most similar relevance centroid, as follows.

$$R(t) = \underset{i}{\operatorname{argmax}}\, \phi(\vec{e_t}, \vec{C_i}) \tag{3.2}$$

where ϕ represents the similarity function used to measure the closeness of the test essay and the relevance centroids (i.e., cosine similarity). The complete algorithm of our proposed method, *ProtoGR* (Prototypes for Graded Relevance), is outlined in Algorithm 1.

Algorithm 1 Pre-trained *ProtoGR* Scoring

```
 1: function TRAIN(Dataset, Dense Encoder)                          ▷ Train the model
 2:     relevance_levels ← distinct relevance levels in Dataset
 3:     C ← []                                                       ▷ Initialize centroids list
 4:     for $i$ in relevance_levels do                               ▷ For each relevance level
 5:         $S_p^i$ ← essays in Dataset with relevance level i
 6:         $\vec{e_i}$ = encoder($S_p^i$)                          ▷ Encode essays
 7:         $\vec{c_i}$ = mean($e_i$)                               ▷ Compute centroid for level i
 8:         C.append($\vec{c_i}$)                  ▷ Append the centroid to the list of centroids
 9:     return $C$
10: function PREDICT($\vec{e_t}$, C)                                ▷ Prediction function
11:     $R(t) = \operatorname{argmax} \phi(\vec{e_t}, \vec{C})$     ▷ Find closest level
12:     return $R(t)$
```

As mentioned earlier, our method relies heavily on the effectiveness of the encoder model used to represent the essays. Dense retrieval models are originally trained for relevance tasks, thus are perfect option to serve our purpose in scoring the *relevance* trait. It is important to note that, although we redefine the problem as a retrieval problem, this method is not restricted to retrieval encoders; models trained on similar tasks, such as text similarity, are also valid options. We discuss the encoder selection in Chapter 4.

3.1.2. Fine-tuning for Task-specific Scoring

As we reframe AES relevance scoring as a retriaval task, we also draw inspiration from their fine-tuning strategies, while adapting them to fit the specific context of graded relevance.

In tasks involving retrieval with dense models, the training dataset comprises queries, relevant, and non-relevant documents. The primary objective is to train the model to decide whether a given document is relevant to a specific query. In our context, essays within the same relevance level are considered "relevant to each other," and essays with different relevance levels are considered "non-relevant to each other." Hence, our model needs to be optimized to differentiate between the different relevance levels.

Contrastive loss has been widely adopted in retrieval models for its effectiveness in learning discriminative embeddings [38], [39]. In our context, it serves a dual purpose: clustering essays with the same relevance level while ensuring essays from different levels remain distinct. This approach mirrors retrieval tasks, where the goal is to bring relevant documents closer and push non-relevant ones farther apart in the embedding space.

In our case, the loss function is treated as a hyperparameter, with options including the original loss function (the loss function that the encoder was trained with) and Pairwise Softmax Cross-Entropy (PSCE) loss. While using the encoder's original loss function i.e., the one it was trained with, may seem logical, PSCE loss has proven effective in high-performance dense retrieval models [38] and is also friendly to GPU memory usage compared to other loss functions.

PSCE loss considers the similarity between positive and negative examples relative to an anchor example, enhancing the model's capacity to separate positive from negative instances.

$$\mathcal{L}_{\text{PSCE}}(s_a, s^+, s^-) = -\log \frac{e^{\phi(s_a,s^+)}}{e^{\phi(s_a,s^+)} + e^{\phi(s_a,s^-)}} \tag{3.3}$$

This setup raises multiple alternatives in terms of how the negative training examples are sampled. For an anchor essay example s_a of relevance level $R(s_a)$, we can sample negative examples s^- from all the other relevance levels, i.e., $R(s_a) \neq R(s^-)$. We denote this as sampling from *All* levels. This exposes the model to training triplets from all different levels. We can also restrict the sampling to the relevance levels that are relatively far from the relevance level of the anchor, i.e., $R(s_a) \neq R(s^-) \pm 1$. We denote this as sampling from *Easy* levels, since they should be easy to distinguish. The rationale is that essays with closer scores might actually have the same score if rated by a different rater, potentially confusing the model. Alternatively, we can restrict the sampling to the relevance levels that are closest to the relevance level of the anchor, i.e., $R(s_a) = R(s^-) \pm 1$. We denote this as sampling from *Hard* levels, since they should be hard to distinguish. The rationale is that by mastering this, the model will also be better at distinguishing relevance levels with larger margins. In each case, the number of negative examples we sample from each selected level might vary.

As the encoder here is fine-tuned, we, hereafter, denote such model by **ft-*ProtoGR***, the **fine-tuned** Prototypes for Graded Relevance.

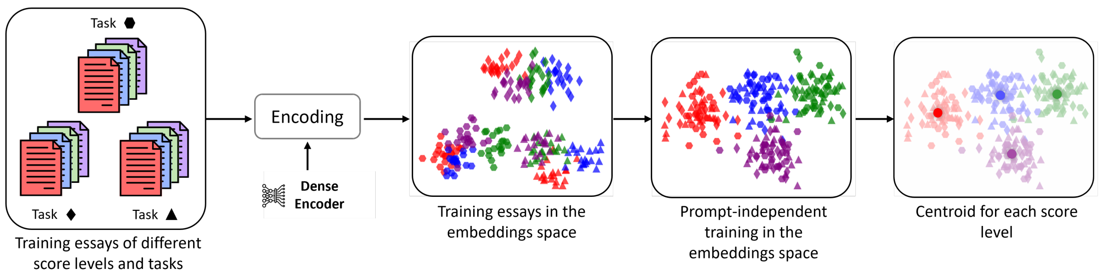

Task ⬢
Task ◆
Task ▲
Training essays of different score levels and tasks
Encoding
Dense Encoder
Training essays in the embeddings space
Prompt-independent training in the embeddings space
Centroid for each score level

Figure 3.2. Training phase of our proposed cross-task approach with task-independent representations.

3.1.3. Cross-Task Scoring

In the earlier scenarios, we assume there are labeled (i.e., scored or graded) essays for the given writing task, which are needed to build a task-specific model, whether it is pre-trained or fine-tuned. However, this is not always available. We might need to build a model with *no* labeled essays for the test task, i.e., a “zero-shot” model. In such a case, the only available labeled essays to learn from come from *other* different tasks than the test task, hence denoted as the *cross-task* scenario. In the cross-task setup, the training typically utilizes multiple *source* tasks, whereas testing is conducted on an *unseen target* task.

Applying the pre-trained or fine-tuned *ProtoGR* approach directly to the cross-task scenario (by just training on the superset of all the training essays of the source tasks) would yield separate clusters (in the embedding space) of task-related essays rather than clusters of essays of same relevance level regardless of the task. Hence, the challenge of that scenario in our approach is how to “normalize” the representation of the essays in order to get “task-independent” representations.

Since the dense encoder represents texts in an embedding “semantic” space, it is tempting to make those representations *task-independent* by simply subtracting the task-prompt representation, disentangling the topic-specific semantic features from the relevance level features. Consequently, the cluster of each relevance level will contain the normalized task-independent essay embeddings from all the source tasks. Accordingly, the computed centroids are potentially task-independent. Similar to the earlier scenario, $1NN$ is used to classify the normalized essays for the target task.

Formally, we are given a set of source tasks P, each with a task-prompt p_n and a corresponding set of labeled essays S_{p_n}. For a relevance level i, the corresponding centroid $\vec{C_i}$ is computed as:

$$\vec{C_i} = \frac{1}{\sum_{p_n \in P} |S^i_{p_n}|} \sum_{p_n \in P} \sum_{s \in S^i_{p_n}} \vec{e}_s - \vec{p}_n \tag{3.4}$$

The relevance level $R(t)$ for a test essay t is then computed using Equation 3.2 except that $\vec{e}_t$ is now normalized as $\vec{e}_t - \vec{p}_t$.

This approach scores the essay based on the aggregated information of the relevance levels from the source tasks. Nevertheless, an essential piece of information that can also be considered is the similarity with the target task-prompt. Consequently, we employed both the scores determined by the nearest centroid and the similarity with the target task-prompt to assign the final relevance score of the essay. For a test essay t

and a target task-prompt p' that has a maximum relevance score of $r_{\max}$, the similarity score $S(t)$ is normalized as follows:

$$S(t) = \phi(\vec{e}_t, \vec{p'}) * r_{\max} \tag{3.5}$$

where ϕ is the similarity function with a $[0 - 1]$ output range. Then, the final score $R^*(t)$ of the test essay t is computed as the average between the two scores:

$$R^*(t) = \frac{1}{2}(R(t) + S(t)) \tag{3.6}$$

The fine-tuning process in the cross-task scenario follows the same setup for the task-specific scenario. The only difference lies in the essays' representations, which are made task-independent by subtracting the embedding vector of the task-prompt from the anchor, positive, and negative examples during the fine-tuning process.

Our general approach for the cross-task scenario is hereafter denoted by a prefix **ct** amended to the pre-trained or fine-tuned model names for Graded Relevance.

With that being said, in the chapter we discussed our approach in two different settings, task-specific and cross-task. The next chapter discuss how we applied *ProtoGR* to English essays in Chapter 4 and Arabic essays in Chapter 5.

CHAPTER 4: EXPERIMENTAL EVALUATION ON ENGLISH ESSAYS

4.1. Experimental Setup

In this section, we introduce the setup used to conduct our experiments for the English dataset. We first discuss the encoder selection criteria (section 4.1.1). Then, we present the dataset we used for evaluation (section 4.1.2) and the specific evaluation measure for the AES task (section 4.1.3). We next discuss the hyper-parameters we used for fine-tuning (section 4.1.4), before we list the baselines with which we compare our work for the task-specific (section 4.1.5) and cross-task (section 4.1.6) approaches.

4.1.1. Encoder Criteria

As previously discussed, selecting an appropriate encoder is crucial, as the scoring mechanism depends on the essay's representation. To evaluate the impact of various encoders on our task, we included a diverse set of dense retrieval and non-retrieval encoders for comparison.

Several criteria guided our encoder selection. First, we included encoders from three distinct categories: retrieval, text similarity, and universal embeddings. Second, within each category, we prioritized models that performed well in their respective areas. Finally, we also considered models that performed well on the Hugging Face MTEB leaderboard[1]. With that being said, the encoders selected from the retrieval category are: Contriever, BGE and BGE-M3. The encoders selected from text similarity are SimCSE, GTE and E5, and finally, the encoders selected from the universal encoders are: BERT and RoBERTa. These chosen pre-trained models were then tested on our task using the proposed approach, after that we selected the top-performing models for further analysis.

4.1.2. Dataset

We employed the commonly-used Automated Student's Assessment Prize (ASAP)[2] and ASAP++ [6] datasets. ASAP dataset comprises 8 different tasks T, where each task has a set of written essays. The original ASAP dataset contains the holistic scores for all tasks and the trait scores only for T7 and T8. ASAP++ extends ASAP by scoring the essay traits for tasks T1-T6. To test our approach, we used the tasks that have annotations for the relevance trait, namely, T3, T4, T5, and T6. Table 4.1 summarizes the dataset we used in our experiment. Following previous work, for fair comparison, we use 5-fold cross-validation with the same folds used by Taghipour and Ng [40] for task-specific models.

Table 4.1. ASAP++ relevance trait tasks.

Task	Relevance Levels	Ave. Length (words)	#Essays
T3	0-3	100	1726
T4	0-3	100	1772
T5	0-4	125	1805
T6	0-4	150	1800

[1] `https://huggingface.co/spaces/mteb/leaderboard`

[2] `https://www.kaggle.com/c/asap-aes`

4.1.3. Evaluation Measure

For evaluating our approach, we used Quadratic Weighted Kappa (QWK) [41], which is a widely used measure for AES. QWK measures the agreement between the scores of two raters, in our case, the human rater and the system. QWK is suitable for this task as it weighs the degree of disagreement between the raters, which is what we want as the scores are *ordered*.

4.1.4. Implementation and Hyper-parameters

All of our experiments are carried out with the Pytorch library.[3] We use the available checkpoint accessible on Hugging Face's model hub for Contriever[4], BGE[5], BGE-M3[6], SimCSE[7], GTE[8], E5[9], BERT[10] and RoBERT[11].

For the hyper-parameter settings, we used the AdamW optimizer with a fixed learning rate of 1e-6 and a batch size of 16 for all models, except for BGE-M3, where we reduced the batch size to 8 due to limited GPU memory. When using 5 negative examples per score level, the dataset size increases, hence, we evaluate every 100 steps, and apply early stopping with a patience of 10 epochs (i.e., 1,000 steps). Otherwise, we evaluated at the end of each epoch and applied early stopping with a patience of 10 epochs. For the cross-prompt setting, we use the same hyper-parameters. However, we randomly select one task as the development set to tune the number of epochs, then retrain using all source tasks with the tuned epoch setting.

4.1.5. Task-specific Baselines

We employed the following baselines based on their approach diversity and their performance:

- *Feature-based*: Mathias and Bhattacharyya [3] utilized multiple models initially designed for holistic scoring. One of their baselines is a feature-based model described in [6].

- *Attention-based Neural Model* [3]: This model, which is based on the work of Dong, Zhang, and Yang [7], achieved the best results for the relevance trait. Hence, we use it as the SOTA baseline.

- *Multi-task Neural Model* [10]: This work developed a multi-task setup for holistic scoring by scoring the traits as sub-tasks. In a specific experiment, individual traits were set as the primary task for prediction, utilizing the other traits alongside the holistic score as sub-tasks. The reported results provided an average score across all tasks. As such, our comparison will be based on this averaged metric.

[3]`https://pytorch.org/`
[4]`https://huggingface.co/facebook/contriever`
[5]`https://huggingface.co/BAAI/bge-base-en-v1.5`
[6]`https://huggingface.co/BAAI/bge-m3`
[7]`https://huggingface.co/princeton-nlp/sup-simcse-roberta-base`
[8]`https://huggingface.co/thenlper/gte-base`
[9]`https://huggingface.co/intfloat/e5-base-v2`
[10]`google-bert/bert-base-uncased`
[11]`https://huggingface.co/FacebookAI/roberta-base`

- *LLM-based Model*: This work is one of the few successful applications of LLMs in AES [28]. The authors fine-tuned the T5 model for task-specific scoring. However, as mentioned earlier, their model was trained on all tasks combined, which does not algin with the task-specific setting.
- *RL-based Model*: The authors in [28] extended their work to use QWK as the optimization metric using RL[30]. The training data issue, however, persist in this baseline as well.

4.1.6. Cross-task Baselines

We adopted the following baselines based on the same criteria of task-specific baselines:

- *Vanilla Baseline*: To test the idea of removing topic information by subtracting the encoded task-prompt from the encoded essays, we developed a baseline model that uses the same approach for cross-task setup but without removing the topic information from the essays' representations. We denote this baseline as the "vanilla" cross-task approach **ct**^{v}-**pt**-*ProtoGR*.
- *PMAES* [18]: This recent work developed a prompt-mapping contrastive learning strategy to capture the shared information between the source and target tasks. To score the traits, a different output layer is used for each trait.
- *ProTACT* [17]: This work proposed a shared model to learn the prompt-aware essay representations. Subsequently, trait-specific layers were introduced on top of the shared layers to score the individual traits.
- *Feature-based* [32]: This work utilized different set of features with a simple neural model achieving the best performance on the relevance trait specifically.

4.2. Results and Discussion

In this section, we present and discuss the results of our experiments addressing the four research questions. We first illustrate the performance of the **pt**-*ProtoGR* model comparing it with SOTA baselines. Then discuss the fine-tuning process and results. Next, we discuss the performance of **pt**-*ProtoGR* and **ft**-*ProtoGR* in few-shot settings. Finally, we show cross-task scenario performance.

4.2.1. Effectiveness of Out-of-the-Box Pre-trained Encoders on English Essays (RQ1)

4.2.1.1. ***Encoders Evaluation***

The first stage is to choose strong encoders for further analysis. As previously mentioned, we chose encoders from three categories: dense retrieval, text similarity, and universal encoders. Table 4.2 presents these selected encoders and their respective results on the development set. Notably, almost all encoders demonstrated strong performance, which was unexpected given that these models were not trained on student essays. This observation raises an intriguing question: Is this strong performance due to the relevance trait we are scoring, potentially aligning with relevance-related tasks

Table 4.2. The selected encoders from each category and their **relevance** trait QWK performance on the **development set**.

Model type	**Model**	**T3**	**T4**	**T5**	**T6**	**Ave**
dense retrieval	Contriever	0.624	0.667	0.608	0.661	**0.640**
	BGE	0.534	0.595	0.527	0.560	0.554
	BGE-M3	0.658	0.654	0.602	0.637	0.638
Text similarity	SimCSE	0.643	0.688	0.661	0.648	**0.660**
	GTE	0.516	0.584	0.491	0.542	0.533
	E5	0.598	0.662	0.536	0.610	0.602
Universal encoder	BERT	0.597	0.679	0.572	0.611	0.615
	RoBERTa	0.640	0.675	0.634	0.529	**0.619**

these models were originally trained on, or does it stem from a combination of method and encoder quality?

To explore this question, we selected a distinct trait less aligned with relevance-based models, namely, the organization trait. Table 4.3 displays the results. As anticipated, the performance on organization is weaker than on relevance, likely because these models were not trained to capture the structural aspects of a text. However, it is interesting to observe that universal encoders, specifically BERT and RoBERTa, which were not designed for relevance tasks, outperformed others. Furthermore, they were followed by their fine-tuned versions, Contriever and SimCSE, respectively.

Table 4.3. Evaluating the pre-trained encoders on the **Organization** trait. The reported values are QWK performance on **development set**.

Model type	**Model**	**T1**	**T2**	**T7**	**T8**	**Ave**
dense retrieval	Contriever	0.520	0.556	0.490	0.226	0.448
	BGE	0.334	0.369	0.393	0.070	0.291
	BGE-M3	0.413	0.417	0.540	0.092	0.365
Text similarity	SimCSE	0.458	0.476	0.589	0.152	0.419
	GTE	0.389	0.402	0.395	0.129	0.329
	E5	0.429	0.475	0.486	0.160	0.388
Universal encoder	BERT	0.520	0.583	0.557	0.228	0.472
	RoBERTa	0.601	0.572	0.554	0.255	0.496

This simple experiment underscores two important considerations for researchers designing an AES system. First, each trait, especially distinct ones, should ideally be modeled independently, as different traits may require different representations and strategies. Second, it is crucial to carefully select the method for representing essays, especially when using pre-trained models. Certain models may be inherently better suited to capturing specific aspects of writing, and a model's original training can strongly influence its effectiveness on different traits. Future research might focus on evaluating the suitability of various pre-trained models for each trait and developing tailored approaches for each trait.

With that in mind, moving forward, we selected the best-performing models, namely, SimCSE, Contriever, and BGE-M3, for further analysis. Additionally, we also test BERT and RoBERTa to compare them with their respective fine-tuned version Contriever and SimCSE.

4.2.1.2. ***Performance Against SOTA Models***

We test the selected encoders on the test set to compare their performance with other state-of-the-art models. Table 4.4 compares the encoders against the baseline models. Remarkably, this simple and straightforward idea achieved an average QWK of 0.661 with an *out-of-the-box* SimCSE model, which is quite effective for the AES task. It even performs consistently across tasks. Furthermore, it clearly outperforms the feature-based baseline, with about 9-points lag behind the SOTA model.

Table 4.4. Performance (in QWK) of our pre-trained *ProtoGR* on the **test** sets, compared to the baselines. Baseline results are reported from their respective papers.

Model	**T3**	**T4**	**T5**	**T6**	**Ave.**
Feature-based [3]	0.575	0.636	0.639	0.581	0.608
Multi-task NM [10]	-	-	-	-	0.730
Attn-based NM [3]	0.683	0.738	0.719	0.783	0.731
LLM-based [28]	-	-	-	-	0.751*
RL-based [30]	-	-	-	-	0.754*
pt-Contriever	0.633	0.673	0.633	0.693	0.658
pt-BGE-M3	0.657	0.647	0.598	0.636	0.635
pt-SimCSE	0.642	0.685	0.658	0.659	0.661
pt-BERT	0.593	0.678	0.573	0.607	0.613
pt-RoBERTa	0.640	0.674	0.633	0.534	0.620

As observed before, all five encoders performed fairly well, suggesting that their embedding spaces are effective for this task. This performance highlights the versatility of these encoders and underscores the potential of our proposed framework.

4.2.2. Effect of Fine-tuning Dense Encoders on English Essays (RQ2)

With the encouraging performance of the pre-trained models shown in the previous experiment, we then turn to examining the performance of these models after fine-tuning.

4.2.2.1. Fine-tuning Process of ProtoGR *on English Essays*

For fine-tuning our models, we have 4 hyper-parameters: the similarity function, the loss function, the selection criteria of negative training samples, and the number of negative training samples per relevance level. As it is impractical to tune those hyper-parameters for each model together with a grid search, we tune one at a time while fixing the others in a sequential series of experiments A through D.

Additionally, we make the following assumption: the similarity function, selection criteria for negative training samples, and the number of negative training samples per relevance level are model-independent. In other words, once an optimal setting for these hyper-parameters is found for one model, we apply the same configuration to all other models. However, the loss function is expected to be model-dependent and can significantly affect the performance of the model, so it is tuned separately for each model in experiment D.

Table 4.5 shows all experiments with different variations conducted with *Contriever* on the development sets and using the PSCE loss function. After tuning model-independent hyper-parameters, we tune the loss function with the best configuration from the previous step for all models.

Table 4.5. Contriever performance (in QWK) of different variations of **ft**-*ProtoGR* on the **development** sets. S_l stands for sample per score level.

Configuration				**Performance**				
Exp.	**Similarity fn.**	**Neg. Sampling**	**Neg.** S_l	**T3**	**T4**	**T5**	**T6**	**Ave.**
A	Cosine	All levels	1	0.718	0.763	0.732	0.789	0.751
	Euclidean	All levels	1	0.710	0.715	0.798	0.727	0.737
B	Cosine	Easy levels	1	0.675	0.727	0.788	0.724	0.728
	Cosine	Hard levels	1	0.658	0.683	0.665	0.659	0.666
C	Cosine	All levels	2	0.720	0.760	0.792	0.738	0.752
	Cosine	All levels	3	**0.726**	0.766	**0.801**	0.737	0.758
	Cosine	All levels	4	0.722	**0.775**	0.797	0.738	0.758
	Cosine	All levels	5	**0.726**	0.774	0.739	**0.801**	**0.760**

Firstly, we tested two similarity functions, cosine similarity and Euclidean distance, selecting negative samples from all relevance levels, and choosing only 1 negative sample per relevance level. Cosine similarity is a metric frequently utilized for measuring the similarity between two vectors irrespective of their magnitudes. In contrast, Euclidean distance calculates the straight-line distance between two points, offering a different perspective on the similarity. The results of experiment A show the superiority of the cosine similarity function, so we use it in the rest of the experiments.

We then examine, in experiment B, the effect of the other ways of selecting negative samples (Easy and Hard levels), which differ in what relevance levels to choose the samples from. The results revealed that using Easy negative samples exhibited better performance, as the model was better able to learn from them to distinguish between the different relevance grades; however, combining both Easy and Hard levels (which comprise the All levels option, shown in experiment A) was even better, indicating that they are complementary and both are needed to learn a better model, and highlighting the benefit of providing the model with contrastive examples from various levels.

Experiment C examines the effect of training with different numbers of negative samples per relevance level. As expected, the more negative samples we include, the better the performance. However, the performance was slightly improving from 0.751 (with one sample per level) to 0.76 (with 5 samples per level). While this points to the importance of increasing the size of our training set, increasing it further incurs further cost in terms of training time and computing resources.

Given the above findings, we settle on the following configuration for the final **ft**-*ProtoGR* model: cosine similarity as the similarity function, selecting negative samples from all relevance levels, and drawing 5 negative samples per each level.

After identifying the best model-independent hyperparameters, we tested two loss functions for each encoder: its original loss function and the PSCE loss function. Table 4.6 shows the results on the development set for the three encoders. Surprisingly, PSCE loss consistently performed better than the model's original loss. This discrepancy may be closely tied to the nature of our problem reformulation. The later loss function has shown effectiveness for retrieval tasks, where the goal is to rank relevant documents higher than non-relevant ones. This aligns better with our problem setup.

Table 4.6. **Experiment D** results on development set of tuning the loss function for the three encoders.

Loss	**Model**	**T3**	**T4**	**T5**	**T6**	**Ave.**
Original Loss	Contriever	0.658	0.693	0.710	0.668	0.682
	SimCSE	0.662	0.704	0.688	0.684	0.684
	BGE-M3	0.741	0.786	0.717	0.790	0.758
PSCE	Contriever	0.726	0.774	0.739	0.801	**0.755**
	SimCSE	0.749	0.806	0.743	0.792	**0.773**
	BGE-M3	0.729	0.776	0.748	0.786	**0.760**

*4.2.2.2. Performance of **ft**-*ProtoGR *Against SOTA Models*

Table 4.7 shows the performance of **ft**-*ProtoGR* on the *test* sets, contrasting it with the baselines. While we report the performance of LLM-based [28] and RL-based [30] models, we do not consider them as baselines since the comparison is not entirely fair. As mentioned earlier, Do, Kim, and Lee [28] and Do, Ryu, and Lee [30] used all tasks in their training sets, which does not align with the literature's definition of task-specific scoring.

BGE-M3, SimCSE, and RoBERTa achieved an average score of 0.755, outperforming the previous SOTA model by nearly two points and establishing a new SOTA for the problem of graded relevance scoring of written essays. Moreover, they were also on par with the baselines presented in [28], [30].

Several interesting conclusions can be drawn from this table. First, BERT and its retrieval-model fine-tuned version, Contriever, show almost identical performance. The same could be said about RoBERTa and SimCSE. This suggests that regardless of their previous training, these models converged to similar results after fine-tuning and weight adjustment, even though Contriever initially outperformed BERT when tested in its pretrained state. Second, contrastive learning proved to be effective, as **ft**-*ProtoGR*-BERT outperformed regressor-BERT.

These findings encourage utilizing dense retrieval and similarity-based models for downstream non-retrieval tasks, wherein a simple problem reformulation with 1NN can achieve impressive performance.

Our best experimented setup is constrained by allowing only up to 5 negative samples per relevance level per anchor essay. However, it is worth noting that we have

Table 4.7. Performance (in QWK) of **ft**-*ProtoGR* on the **test** sets, compared to the baselines.

Model	**T3**	**T4**	**T5**	**T6**	**Ave.**
Feature-based [3]	0.575	0.636	0.639	0.581	0.608
Multi-task NM [10]	-	-	-	-	0.730
Attn-based NM [3]	0.683	0.738	0.719	0.783	0.731
Regressor-BERT	0.676	0.709	0.688	0.776	0.712
LLM-based [28]	-	-	-	-	0.751*
RL-based [30]	-	-	-	-	0.754*
ft-*ProtoGR*-Contriever	0.704	0.766	0.707	0.785	0.741
ft-*ProtoGR*-BGE-M3	0.734	0.777	0.727	0.781	**0.755**
ft-*ProtoGR*-SimCSE	0.735	0.779	0.731	0.776	**0.755**
ft-*ProtoGR*-BERT	0.718	0.758	0.720	0.775	0.743
ft-*ProtoGR*-RoBERTa	0.728	0.783	0.716	0.794	**0.755**

not explored the optimal scenario of including all possible negative examples for each anchor essay. We argue that doing so could potentially achieve even greater performance improvements.

4.2.2.3. Visualizing ProtoGR *on English Essays*

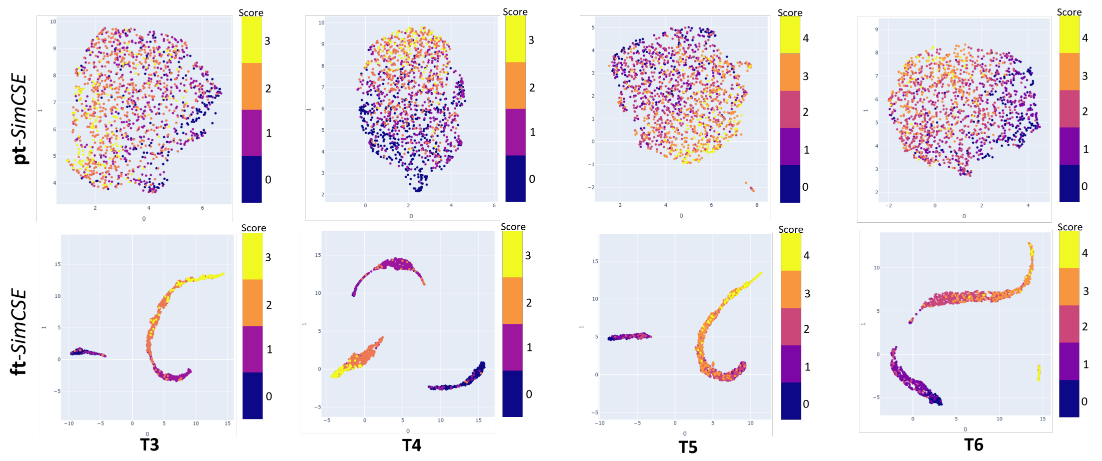


Figure 4.1. SimCSE 2D Visualization of the training set of each task, fold 0, before and after fine-tuning using UMAP.

In Figure 4.1, we illustrate the impact of fine-tuning SimCSE. These visualizations pertain to the training set of each task. Employing the 2D UMAP dimensional reduction method [42], we plotted the encoded essays in a scatter plot for analysis. The figures at the top show how the pre-trained SimCSE encodes the essays. Despite a somewhat mixed arrangement of essays with varying scores, distinct groups corresponding to each relevance level are somewhat discernible. However, after fine-tuning, a clearer ordinal hierarchy emerges between relevance levels, with distinct clusters forming for each

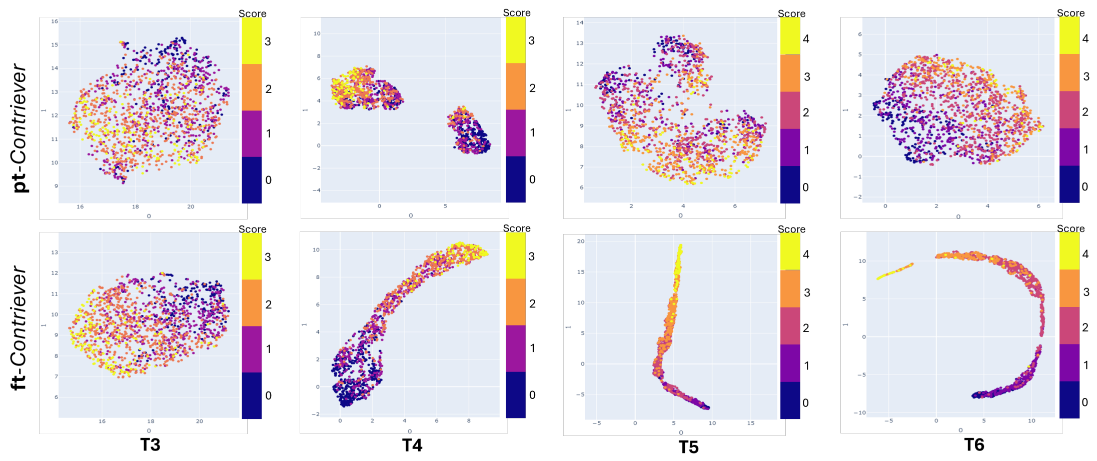


Figure 4.2. Contriever 2D Visualization of the training set of each task, fold 0, before and after fine-tuning using UMAP.

score. This refinement is particularly prominent in tasks 4 and 6. Notably, essays with a score of 0 consistently form their own distinct cluster, making them more distinguishable compared to scores 1 and 2. Finally, it is observable that the fine-tuning process not only enhances score-level distinctions but also contributes to the compactness of each cluster of essays.

It was also interesting to observe how the lowest-performing model, Contriever, represented the essays and compare it to the best-performing model. Figure 4.2 shows the essay representations after fine-tuning. Notably, Contriever was able to establish an ordinal hierarchy between relevance levels. However, the clusters are not as well-separated as those produced by SimCSE, which could explain Contriever's lower performance.

4.2.2.4. Error Analysis

Figure 4.3 the confusion matrices for SimCSE predictions, highlighting patterns of prediction accuracy and errors. Overall, the model demonstrates the ability to distinguish essays with extreme scores, particularly score 0, which consistently forms well-defined clusters with minimal confusion. However, intermediate scores, such as 1 and 2, exhibit more overlap, leading to notable misclassifications between adjacent levels. This also aligns with the observations from figure 4.1. Additionally, this pattern is reflected in the F1 scores for each score level across different tasks, as summarized in Table 4.8.

Table 4.8. F1 score for each score level across the different tasks using SimCSE

	S0	S1	S2	S3	S4
T3	0.68	0.48	0.62	0.49	-
T4	0.81	0.47	0.54	0.50	-
T5	0.58	0.43	0.53	0.43	0.44
T6	0.71	0.47	0.67	0.43	0.32
Ave	0.70	0.46	0.59	0.46	0.38

Upon further investigation, we hypothesize that the confusion between scores 1 and 2 could be due to the ambiguity in their descriptions. Specifically, the criteria for these scores might be unclear for humans. The descriptions are as follows:

Score 2: The response shows a good understanding of the meaning of the text and question, and occasionally wanders off topic.

Score 1: The response shows a misreading of the text or question, or consistently wanders off topic.

Both score levels can include responses that ”wander off topic;” however, it is unclear what degree of wandering corresponds to a score of level 1 or 2. In contrast, score 0 and the highest score (i.e., score 3 or 4) are more easily distinguishable. Score 0 represents a completely irrelevant response, while the highest score indicates a response of exceptional quality.

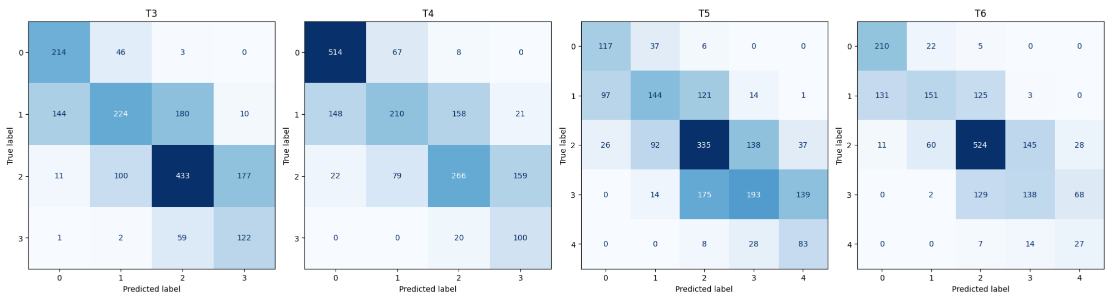


Figure 4.3. Confusion matrix of **ft**-*ProtoGR*-SimCSE predictions of the **test** set.

While score level 0 was the most distinguishable overall, an interesting pattern emerged where score level 2 performed exceptionally well. This led us to examine the distribution of score levels in the training set, as shown in Table 4.9. Indeed, in nearly all tasks, score level 2 represented the highest percentage of the data. The issue of data imbalance has not been addressed in previous AES research, as much of the work treats the problem as a regression task. However, this observation suggests that data imbalance could be limiting further performance improvements.

Table 4.9. Data distribution for the relevance trait in fold 0.

	S0	S1	S2	S3	S4
T3	13.7%	33.8%	42.5%	9.90%	-
T4	32.9%	31.2%	28.4%	7.40%	-
T5	8.10%	21.1%	34.7%	29.2%	6.80%
T6	12.6%	24.7%	41.5%	18.8%	2.40%

From the above analysis, we encourage future research that explores advanced loss function designs, develops score-difference-aware training strategies, or investigates the impact of data imbalance in the AES problem to address these specific challenges. Additionally, it is crucial to emphasize that developing a robust AES model requires a clear and concise rubric created by experts. Without such a rubric, score levels may become ambiguous for evaluators, leading to assessments driven by subjective feelings rather than objective criteria.

4.2.3. Few-shot Learning

In earlier scenarios, we assume many *labeled* essays from the same task are available (more than a thousand in the ASAP++ dataset). While this is a common setup, it is indeed *impractical* in educational contexts and poses a considerable challenge, as it requires time-consuming manual grading for any new task. *Few-shot* learning alleviates this constraint by requiring only a few graded essays to effectively train the model, making our approach more feasible.

To that end, we study the performance of our pre-trained and fine-tuned models when trained within a k-shot setup, where k is the number of labeled essays per relevance level that are available for training for a given task. We varied k from 5 to 30 with a step of 5. As the number of shots increases, we augmented the initial set by randomly sampling additional 5 essays per relevance level, creating comparable subsets across experiments. We then test the trained models on the standard test set of the task. To improve robustness and mitigate the influence of chance, we repeated the entire process 5 times and reports the average performance over the 5 runs. For this experiment, we employed SimCSE model.

Figure 4.4 illustrates the performance of both **pt**-*ProtoGR* and **ft**-*ProtoGR* with the few-shot setup compared to the full-shot setup (i.e., training with *all* labeled examples).

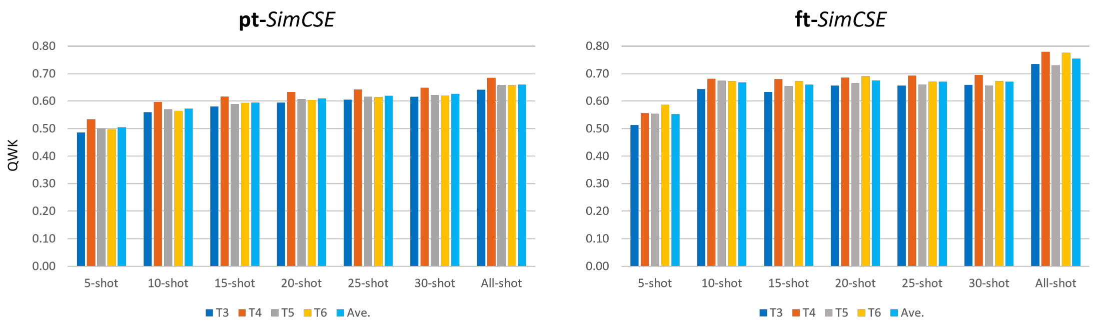


Figure 4.4. Performance of our models with the k-shot setup compared to the full-shot setup on the **test** sets.

As anticipated, a general observation reveals that increasing k positively correlates with improved performance, and the fine-tuned models generally outperform their pre-trained counterparts.

Most notably, the average performance of **ft**-*ProtoGR* with 30 shots trails by approximately only 9 points compared to the full-shot setup; it is crucial to consider that 30 shots imply only 150 labeled essays (assuming 5 relevance levels), which constitutes less than 15% of the full training set that consists of 1000+ essays, reducing the labeling cost by more than 85% while sacrificing about 9% of the performance. This clearly has the potential to control the trade-off between practicality and performance; it indeed opens the door for future efforts toward narrowing the performance gap.

Interestingly, we also noticed that fine-tuning with above 10 shots per relevance level was sufficient to reach a similar performance as the pre-trained model when trained with the full set. This highlights the power of the our model when fine-tuned with few shots.

All in all, our models yielded promising performance in the few-shot scenario, though the rule of thumb of having more data for better performance remains. Nevertheless, our models showcase commendable overall performance even with a fraction of the original data, emphasizing the practicality of our approach.

4.2.4. Cross-Task Scoring (RQ3)

What if we do not have any labeled essays for the test task? Can we leverage labeled essays for other tasks to learn a model that can be used to score essays written for a *new* task for which we do not have any labeled essays? This is exactly the *cross-task* scenario.

For this scenario, the setup goes as follows. For each target (i.e., test) task in the dataset, the 3 other tasks (along with their labeled essays) are used for training. Due to the discrepancy of relevance levels across tasks (T3 and T4 range from 0 to 3, while T5 and T6 range from 0 to 4), centroids for relevance levels 0 to 3 are computed from all of the 3 training tasks, while the centroid for relevance level 4 is computed only from T5 and/or T6. In these experiments, the training set size increases significantly. Due to our resource limitations, we omitted the BGE-M3 model, as it has a larger size.

We experimented with 3 variants of *ProtoGR* for the cross-task scenario. The first is the *vanilla* cross-task model, denoted by the prefix $\mathbf{ct}^v$, which directly uses the centroids of the essays' vectors as done for the task-specific scenarios. The second is the *task-independent* version denoted by the prefix $\mathbf{ct}^i$, in which we made the essay vectors (thus the centroids) task-independent by semantically "disentangling" their corresponding task-prompts. The third is an extended version of the second that is denoted by the prefix $\mathbf{ct}^s$, in which we also involve the similarity with the target task-prompt when we score a given test essay. We tried the second and third variants with both **pt**-*ProtoGR* and **ft**-*ProtoGR* models. For fine-tuning, we used the same hyper-parameters used in the task-specific experiments.

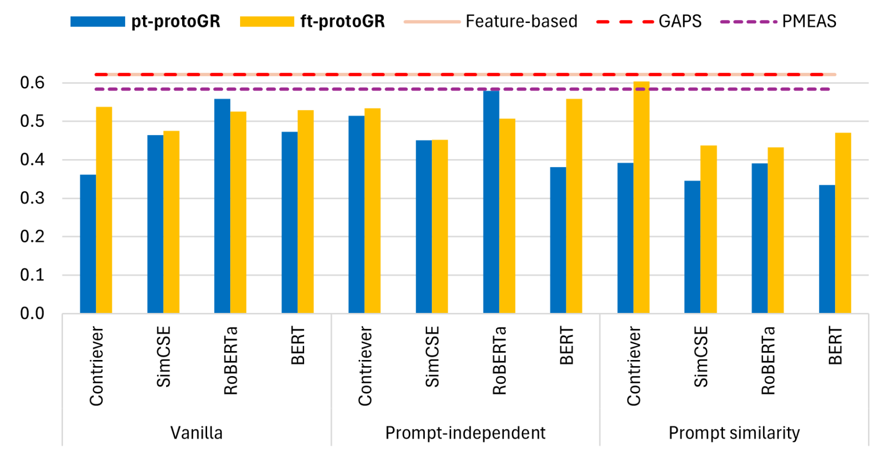


Figure 4.5. Cross-task results on **test-set**. GAPS and Feature-based baseline have the exact reported QWK.

Figure 4.5 shows the performance of our model variants compared with state-of-the-art (SOTA) baselines in the cross-task setup. Several key observations can be made:

pt-*ProtoGR* Setup Within the **pt**-*ProtoGR* setting, ct^i achieved the best performance, followed by ct^v and ct^s. Notably, the **pt**-*ProtoGR*-RoBERTa model performed on par with the PMEAS baseline under the ct^i configuration. The stronger results of ct^i over ct^v in Contriever, SimCSE, and RoBERTa support the idea that removing the "topic" from essays makes them more task-prompt independent, thereby making them better suited to cross-task scenarios.

ft-*ProtoGR* Setup Fine-tuning boosted performance for most models, indicating that the model learned to cluster similarly scored essays together regardless of their specific tasks. However, RoBERTa did not improve, and even decreased in performance, under the ct^v and ct^i configurations, suggesting that not all encoders respond equally to fine-tuning. This underscores the importance of carefully selecting both the encoder and the fine-tuning strategy for a given task. Notably, Contriever surpassed the PMEAS baseline in the ct^s setup. Although Contriever, being a retrieval model, was the least effective in the task-specific setting, its strengths were more evident in cross-task AES.

Despite these findings, none of our models outperformed the GAPS SOTA model. The best-performing variant, ct^s-Contriever, still lagged by two points.

In summary, we evaluated *ProtoGR* in two main settings: task-specific and cross-task. In the task-specific setting, all of our encoders outperformed the SOTA, with three **ft**-*ProtoGR* encoders exceeding the SOTA by 2 points. In the cross-task setting, Contriever outperformed one of the baselines but lagged behind the SOTA model by 2 points. We also observed the impact of **pt**-*ProtoGR* and **ft**-*ProtoGR* when working with a limited number of essays.

In the next chapter, we will present the experimental evaluation on Arabic essays in both task-specific and cross-task setups and compare the results to the baseline models.

CHAPTER 5: EXPERIMENTAL EVALUATION ON ARABIC ESSAYS

5.1. Experimental Setup

This section outlines the experimental framework for the Arabic language. Section 5.1.1 details our encoder selection criteria for the Arabic language. Section 5.1.2 describes the dataset and its division for experiments. Section 5.1.3 details the evaluation measures used. Finally, Sections 5.1.4 and 5.1.5 present the hyper-parameter settings and baseline details, respectively.

5.1.1. Encoder Selection

For Arabic data, due to the limited Arabic-specific encoders, we choose only one encoder from each category, mContriever[1], Multilingual-E5[2], and XLM-RoBERTa[3] from retrieval, text-similarity and universal embeddings encoders respectively. Similar to the English encoders selection criteria, we also choose encoders that perform well on the Hugging Face MTEB leaderboard[4] at the time of our experiments.

5.1.2. Arabic Data

For the Arabic data, we use an *in-house* dataset of 1,275 essays across 8 different tasks, written by native first-year university students under test-like conditions. Each essay is annotated across 7 traits including relevance. The scoring follows a standardized rubric, with the relevance trait assessed on a scale of 0 to 2, with 1-point increments. The annotation process was conducted by two primary Arabic language specialists, with a third annotator available to resolve any disagreements. Annotators were selected with extensive teaching experience and underwent several training sessions to ensure understanding of the rubric and consistency throughout the annotation process. Table 5.1 provides a breakdown of the prompts featured in our Arabic dataset. Figure 5.1 shows the distribution of relevance scores across the eight different tasks.

Table 5.1. Arabic Dataset statistics

Task	Type	Essays	Ave. Len (words)
T1	Explanatory	215	137
T2	Persuasive	210	150
T3	Persuasive	115	501
T4	Persuasive	80	474
T5	Explanatory	184	129
T6	Explanatory	135	132
T7	Persuasive	195	122
T8	Persuasive	141	129

For task-specific scoring, we perform 5-fold cross-validation, using one fold for testing, another for development, and the remaining three for training. The experiment is

[1]https://huggingface.co/facebook/mcontriever

[2]https://huggingface.co/intfloat/multilingual-e5-small

[3]https://huggingface.co/FacebookAI/xlm-roberta-base

[4]https://huggingface.co/spaces/mteb/leaderboard

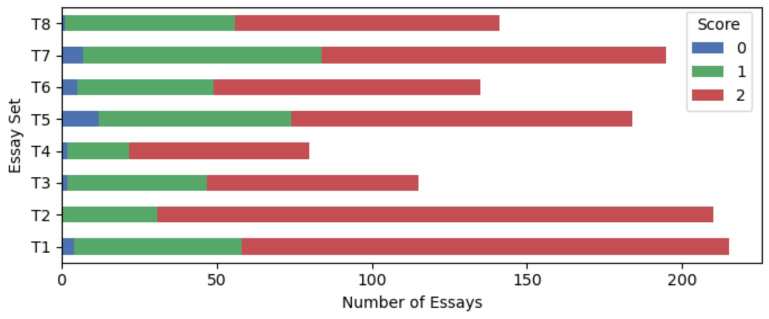


Figure 5.1. Distribution of relevance scores across different tasks in the Arabic dataset.

repeated for each testing fold, and we report the average results across all folds. Similar to the English experiments, hyper-parameters are selected based on the best development set results across all folds and tasks.

For cross-task scoring, we use leave-one-task-out cross-validation, where each task is treated as a test set while the remaining tasks are used for training. For each encoder, we apply the best hyper-parameters obtained from task-specific scoring.

5.1.3. Evaluation Measures

For Arabic data evaluation, we use QWK, the most common metric for assessing agreement between human annotators and systems. However, it has limitations, including a need for a large sample size, and its sensitivity to the score scale [43]. To address these limitations, *in some cases*, we also use the Root Mean Square Error (RMSE), a metric commonly used for ordinal classification tasks [44] similar to predicting score levels in AES tasks. Reporting both QWK and RMSE aims to more reliably evaluate the quality of the tested models.

5.1.4. Implementation and Hyper-parameters

We implement our experiments using the PyTorch library and train the models on Nvidia A10-24Q and Nvidia RTX A6000 GPUs. Optimization is performed using the AdamW optimizer, with a fixed batch size of 16. For hyper-parameter tuning, we conduct a grid search over learning rates ($2e^{-5}$, $3e^{-5}$, and $5e^{-5}$) and the number of trainable layers (3, 6, or none for full fine-tuning), the rest of the hyper-parameters (distance function, number of negative examples per score level) are tuned sequentially. More details are provided in upcoming sections. In task-specific scoring, models are trained for 100 epochs, and the best checkpoint is selected based on the highest QWK score on the development set. For cross-task scoring, we train the models for five epochs and employ the best settings we got from task-specific experiments for each encoder separately.

5.1.5. Baselines

To validate our results, we compare with the following baselines for both task-specific and cross-task scoring:

Table 5.2. Performance (in QWK) of our pre-trained ProtoGR on the Arabic essays **test set**, compared to the baselines.

Model	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**Ave.**
RF	0.079	0.141	-0.121	0.064	0.472	0.437	0.599	0.280	0.244
SVM	0.080	0.230	-0.126	0.070	0.441	0.435	0.605	0.319	0.257
Reg-mContriever	0.107	0.056	0.187	0.214	0.521	0.430	0.617	0.292	0.303
Reg-Multilingual-E5	0.162	0.047	0.230	-0.009	0.606	0.318	0.612	0.149	0.264
Reg-XLM-RoBERTa	0.294	0.113	0.260	0.195	0.465	0.574	0.587	0.118	0.326
pt-mContriever	0.329	0.158	0.321	0.350	0.530	0.456	0.476	0.178	**0.350**
pt-Multilingual-E5	0.329	0.243	0.347	0.181	0.565	0.464	0.607	0.265	**0.375**
pt-XLM-RoBERTa	0.269	0.209	0.186	0.229	0.570	0.563	0.637	0.291	**0.369**

1. **Feature-Based Models:** We use a subset of features from Alqahtani and Al-Saif [23] to train Support Vector Machine (SVM) and Random Forest (RF) models. These features, categorized as surface, syntactic, and lexical, are detailed in Table **??**. Appendix A.1 provides implementation details and describes the hyperparameter tuning process.

2. **Regression Fine-Tuning:** We fine-tune the selected encoders, namely, XLM-RoBERTa, mContriever, and Multilingual-E5, for relevance score prediction by adding a regression head to each model. We refer to these models as Reg-X (e.g., Reg-mContriever). Implementation details and hyperparameter tuning are provided in Appendix A.2.

5.2. Results and Discussion

In this section, we present and analyze the results of our experiments on Arabic essays, addressing the four research questions. We begin by showcasing the performance of the **pt**-*ProtoGR* models in comparison to our baselines. Then, we examine the fine-tuning process and its outcomes. Lastly, we explore their effectiveness in a cross-task scenario.

5.2.1. Effectiveness of Out-of-the-Box Pre-trained Encoders on Arabic Essays (RQ1)

The first research question we address is the performance of the chosen pre-trained encoders. Unlike English, for which we selected multiple encoders, here we chose one encoder from each category, as noted earlier. Consequently, we evaluated them directly on the test set and compared their results with our baselines. Table 5.2 presents **pt**-*ProtoGR* model performance for Arabic essays against feature-based and regression fine-tuned baseline models (Reg-X).

A key observation is the relatively lower scores compared to the performance on English essays (where QWK exceeds 0.7), underscoring the complexity of this data, particularly for the relevance trait. Despite this, all **pt**-*ProtoGR* encoders, out-of-the-box, outperformed our trained baselines by a significant margin. For instance, the top-performing encoder, Multilingual-E5, surpassed the strongest regression fine-tuned baseline, XLM-RoBERTa, by approximately 5 points in QWK and exceeded the strongest feature-based model, SVM, by 11 points.

As noted earlier, QWK has limitations, especially in cases of agreement imbalance [43], which may explain the relatively lower QWK scores. Consequently, we also report the RMSE measure. Figure 5.2 illustrates the performance of the **pt**-*ProtoGR* model and the baselines in terms of RMSE. On average, the **pt**-*ProtoGR* encoders achieved an RMSE of 0.57, indicating that the typical error is only 0.57 points. Interestingly, although the **pt**-*ProtoGR* models consistently achieved higher QWK values, the baselines recorded lower RMSE scores.

This discrepancy can be explained by the different focuses of the metrics. QWK evaluates agreement adjusted for chance and weight errors by their squared distance, emphasizing ordinal consistency. In contrast, RMSE measures the raw magnitude of errors. A model that makes smaller, more frequent errors (e.g., confusing 0 with 1) may exhibit a lower RMSE yet fails to preserve critical ordinal rankings, thereby reducing its QWK. Hence, **pt**-*ProtoGR* makes better ordinal predictions, while maintaining relatively lower RMSE values.

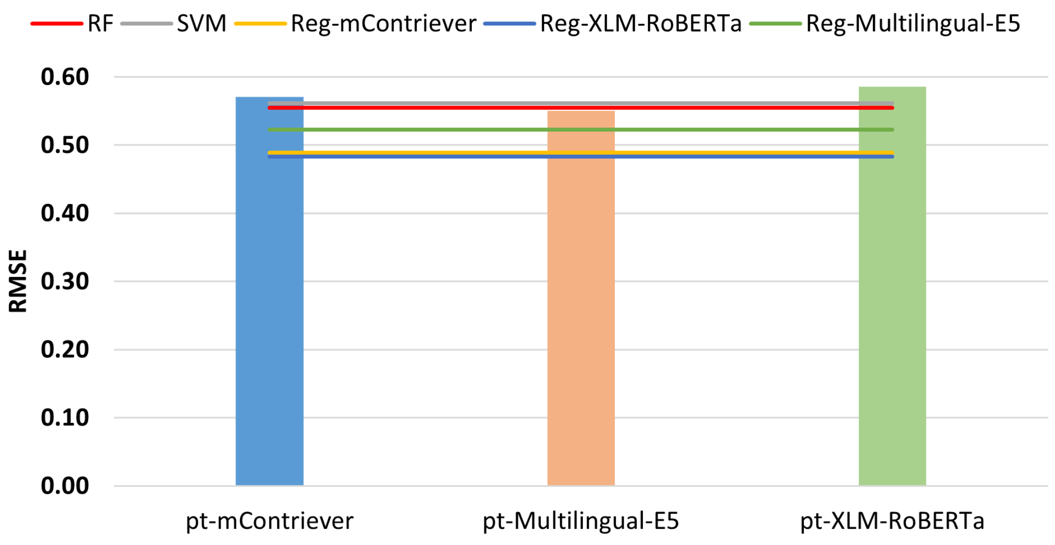


Figure 5.2. Results of **pt**-*ProtoGR* with RMSE as the evaluation metric.

5.2.2. Effect of Fine-tuning Dense Encoders on Arabic Essays (RQ2)

Now, we examine the performance of the **ft**-*ProtoGR* encoders. In this section, we first introduce the fine-tuning process for the Arabic essays, then we show its performance against **pt**-*ProtoGR* and the baselines.

5.2.2.1. Fine-tuning Process of ProtoGR *on Arabic Essays*

For fine-tuning dense encoders on Arabic essays, we tune four hyper-parameters: learning rate, number of trainable layers, similarity function, and number of negative examples per score level. The *type* of negative examples is not applicable to the Arabic relevance trait, as it contains only three score levels. All experiments employ the PSCE loss function consistently.

Table 5.3. **ft**-*ProtoGR* performance with different hyper-parameter settings on the **development** set. S_l stands for sample per score level.

Exp.	**Model**	**Similarity fn.**	**Neg S_l.**	**QWK Ave.**
A	mContriever	cosine	1	0.509
	Multilingual-E5	cosine	1	0.553
	XLM-RoBERTa	cosine	1	0.513
	mContriever	euclidean	1	0.498
	Multilingual-E5	euclidean	1	0.526
	XLM-RoBERTa	euclidean	1	0.554
B	mContriever	cosine	2	0.506
	Multilingual-E5	cosine	2	0.506
	XLM-RoBERTa	euclidean	2	0.552
	mContriever	cosine	3	0.497
	Multilingual-E5	cosine	3	0.525
	XLM-RoBERTa	euclidean	3	0.529
	mContriever	cosine	4	0.501
	Multilingual-E5	cosine	4	0.517
	XLM-RoBERTa	euclidean	4	0.528
	mContriever	cosine	5	0.501
	Multilingual-E5	cosine	5	0.521
	XLM-RoBERTa	euclidean	5	0.528

We conduct a grid search for learning rate and trainable layers using the values $\{$2e-5, 3e-5, 5e-5$\}$ for learning rate and $\{$3, 6, all layers$\}$ for trainable layers. After identifying the optimal combination, we sequentially tune the remaining hyper-parameters.

The grid search yielded the following optimal configurations: a learning rate of 2e-5 for all encoders, 6 trainable layers for Multilingual-E5 and XLM-RoBERTa, and all layers for mContriever. Table 5.3 summarizes Experiment A (identifying the best similarity function for each encoder) and Experiment B (determining the optimal number of negative examples per score level). These experiments were conducted using the best learning rate and layer configurations for each encoder.

For similarity functions, Euclidean distance outperformed cosine similarity for XLM-RoBERTa, while cosine similarity proved superior for mContriever and Multilingual-E5 (Experiment A). Notably, increasing the number of negative examples per score level did not improve performance (Experiment B). Consequently, we retained one negative example per score level for subsequent experiments, pairing Euclidean distance with XLM-RoBERTa and cosine similarity with mContriever and Multilingual-E5.

*5.2.2.2. Performance of **ft**-*ProtoGR *Against Baseline Models*

Finally, we evaluate the fine-tuned encoders on the test set. Figure 5.3 compares their performance against their pre-trained counterparts and baseline models.

Interestingly, unlike the results observed on English essays, **ft**-*ProtoGR* on Arabic essays did not yield significant improvements over the **pt**-*ProtoGR* baselines. This

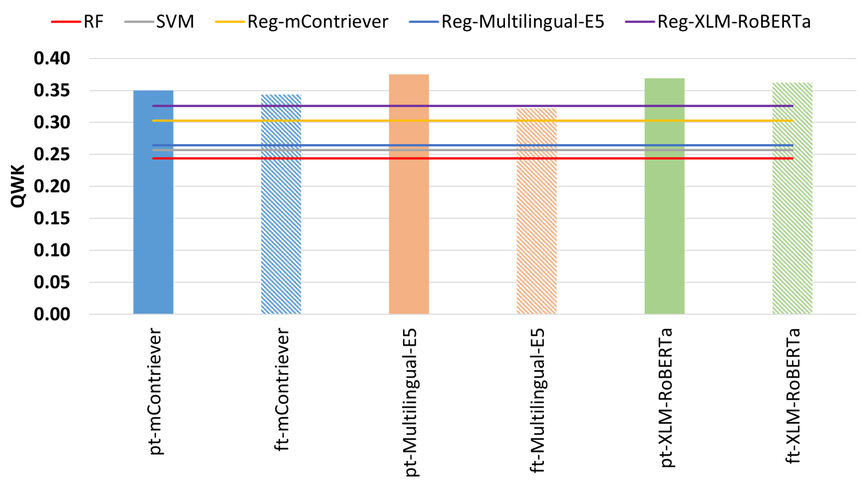


Figure 5.3. **ft**-*ProtoGR* performance on Arabic essays against **pt**-*ProtoGR* and Arabic baselines.

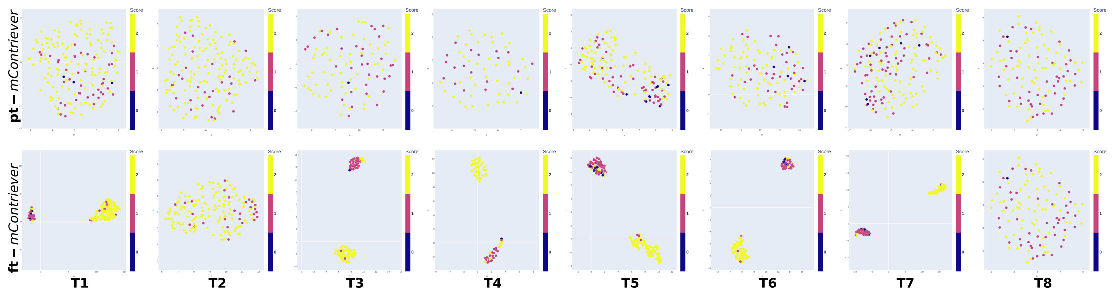
finding is intriguing because the models were explicitly trained on Arabic essays during fine-tuning. The discrepancy may stem from the limited size of the Arabic dataset: while English tasks averaged approximately 1,800 essays per task, Arabic tasks contained only 200 essays per task on average. This smaller dataset size may not provide enough diversity and coverage to effectively capture the complexities needed for robust representation learning.

Furthermore, the marginally higher performance of **pt**-*ProtoGR* models could indicate that their pre-trained multilingual knowledge, acquired from vast and diverse corpora during initial pre-training, provides a stronger foundation for generalization in low-data scenarios. In contrast, fine-tuning on a small Arabic dataset may narrow the model's focus, reducing its ability to leverage broader linguistic patterns. These results highlight two critical insights: First, **pre-trained models can perform relatively well without fine-tuning in certain contexts**. As demonstrated in previous experiments, **ft**-*ProtoGR* significantly outperformed **pt**-*ProtoGR* on English essays, yet this trend did not extend to Arabic. Hence, the size and linguistic nature of the dataset are crucial considerations when deciding whether to fine-tune. Second, **methodological simplicity can be surprisingly effective**. For instance, although Reg-X is fine-tuned, it did not significantly outperform any of our models (whether **pt**-*ProtoGR* or **ft**-*ProtoGR*). By comparison, our approach, particularly **pt**-*ProtoGR*, achieved better results without requiring any additional fine-tuning.

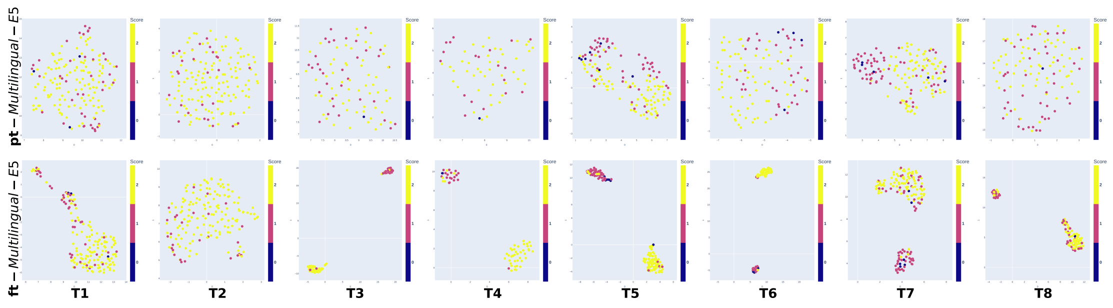
5.2.2.3. *Visualizing* ProtoGR *on Arabic Essays*

Visualizing the encoders in their pre-trained and fine-tuned states can help us get insights on how the encoders represent the essays. Figure 5.4 depicts the encoders in both their pre-trained and fine-tuned states.

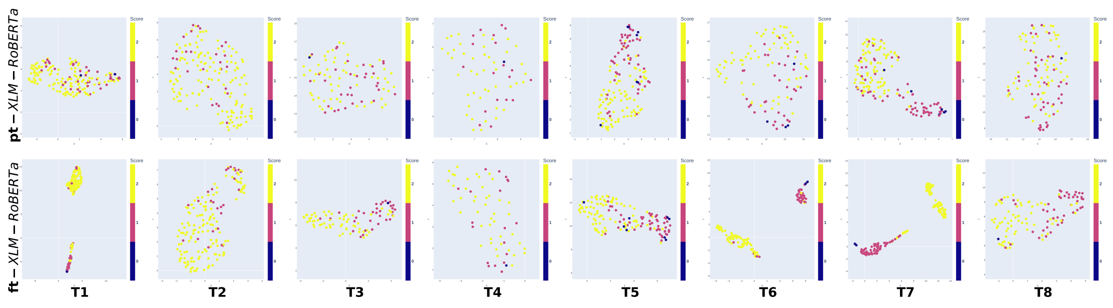
Interestingly, fine-tuning had a similar effect on Arabic essays as it did on English essays. Specifically, there is a clear separation between clusters of different score levels, which is particularly evident in mContriever and Multilingual-E5. However, this separation was not reflected in the quantitative results. This discrepancy might be due to the small number of essays, which may have limited the encoder's ability to capture all



(a) mContriever



(b) Multilingual-E5



(c) XLM-RoBERTa

Figure 5.4. 2D Visualization of the training set of each task in Arabic data set across different encoders using UMAP.

aspects of essays associated with a particular score. Moreover, upon closer examination of the best-performing encoder, pt-Multilingual-E5, a pattern emerges: tasks with more balanced score distributions, specifically T5, T6, T7, and T8, exhibit better QWK performance and more pronounced clusters. In contrast, for highly imbalanced tasks such as T1 and T2, there are no clear clusters, and the encoder achieved relatively low QWK scores. Nevertheless, this visualization encourages further investigation into the lower QWK performance, especially when more Arabic essays become available.

5.2.3. Cross-Task Scoring (RQ3)

We now turn to the cross-task scenario for the Arabic relevance trait AES. Following the methodology applied to English essays, we train each encoder using the optimal hyper-parameters identified in task-specific experiments. We evaluate three variations of scoring functions: vanilla ($\mathbf{ct}^v$), task-independent ($\mathbf{ct}^i$), and task-similarity ($\mathbf{ct}^s$). Figure 5.5 shows the performance of *ProtoGR* against the baselines.

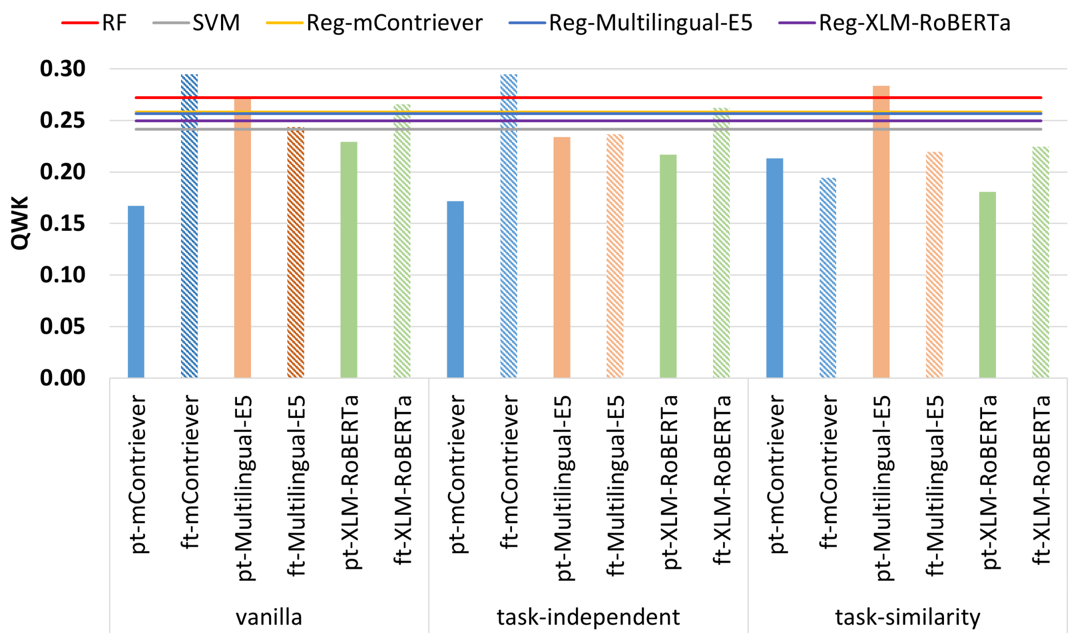


Figure 5.5. Performance of **pt**-*ProtoGR* and **ft**-*ProtoGR* on Arabic essays in the cross-task scenario against the baselines.

For **pt**-*ProtoGR*, the $\mathbf{ct}^s$ scoring function achieves the strongest results with mContriever and Multilingual-E5, outperforming their $\mathbf{ct}^v$ and $\mathbf{ct}^i$ counterparts. Notably, XLM-RoBERTa exhibits distinct behavior, achieving its best performance with the $\mathbf{ct}^v$ scoring function. Interestingly, pre-trained Multilingual-E5 paired with $\mathbf{ct}^s$ surpasses all other pre-trained encoders and the strongest baseline, RF, by a significant margin of 2.1 points.

With **ft**-*ProtoGR*, nearly all variants of our method show marginal performance improvements. For instance, mContriever improves by 13 and 12 points for $\mathbf{ct}^v$ and $\mathbf{ct}^i$, respectively. XLM-RoBERTa demonstrates similar gains, with improvements of 5, 5, and 4 points for $\mathbf{ct}^v$, $\mathbf{ct}^i$, $\mathbf{ct}^s$ respectively. In contrast, Multilingual-E5 shows no improvement with any scoring function after fine-tuning, suggesting encoder-specific optimization requirements.

When comparing **ft**-*ProtoGR* models to baselines, XLM-RoBERTa with $\mathbf{ct}^v$ and $\mathbf{ct}^i$ outperforms nearly all baselines, including its regression-based counterpart, by 2 points. However, mContriever surpasses all baselines, achieving a 2.3-point advantage over the RF baseline.

These results demonstrate the effectiveness of our method in different settings. Specifically, *ProtoGR* achieves SOTA performance using pre-trained Multilingual-E5 with $\mathbf{ct}^s$, while mContriever shows remarkable gains when fine-tuned and outperforms all baselines.

These findings highlight two main priorities for future research: leveraging robust pre-trained encoders such as Multilingual-E5, which excel even with limited task-specific data, and developing encoder-specific fine-tuning strategies.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1. Conclusion

In this study, we introduced a novel approach for graded relevance scoring of written essays that leverages dense retrieval, employing several dense encoders from different categories. For English essays, out of the box, without any fine-tuning, our method yielded promising results, yet post-fine-tuning, it established a new SOTA performance for task-specific scenarios. Furthermore, we proposed a simple adjustment to our approach, eliminating the influence of the task-prompt to enable its adaptability for cross-task settings, achieving a performance that is on par with SOTA baselines. Moreover, we showed that our approach exhibited reasonable performance in more practical scenarios where only few essays are labeled for the target writing task. In particular, with only 30 graded essays per relevance level, we observed about 9% drop in performance while saving more than 85% of the manual labor cost.

For Arabic essays, our method achieves strong out-of-the-box performance, surpassing existing baselines by a 5-point QWK gain. In cross-task scenarios, *ProtoGR* outperforms the strongest baseline by 1 and 2 points when using the pre-trained Multilingual-E5 with the $\mathbf{ct}^s$ scoring function and the fine-tuned mContriever with both $\mathbf{ct}^v$ and $\mathbf{ct}^i$ respectively.

6.2. Publications

6.2.1. Main publications

- Albatarni, S., Eltanbouly, S., & Elsayed, T. (2024, July). Graded relevance scoring of written essays with dense retrieval. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 1329-1338) [45].

6.2.2. Other publications

- Bashendy, M., Albatarni, S., Eltanbouly, S., Zahran, E., Elhuseyin, H., Elsayed, T., Massoud, W., & Bouamor, H. (2024, August). QAES: First Publicly-Available Trait-Specific Annotations for Automated Scoring of Arabic Essays. In Proceedings of The Second Arabic Natural Language Processing Conference (pp. 337-351).

- Mansour, W. A., Albatarni, S., Eltanbouly, S., & Elsayed, T. (2024, May). Can Large Language Models Automatically Score Proficiency of Written Essays?. In Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024) (pp. 2777-2786).

6.3. Future Work

This study paves the way for several promising research directions. First, while our methodology originates from dense retrieval for relevance scoring, its framework can be extended to assess additional essay traits (e.g., vocabulary, style). For English relevance scoring in cross-task scenarios, current performance gaps suggest opportunities for refinement. A potential solution could involve designing task-agnostic loss functions that explicitly model ambiguity and domain shifts across diverse writing prompts. In Arabic relevance trait AES, significant room remains to enhance model generalizability. Future efforts might focus on optimizing encoder fine-tuning strategies for low-resource languages or incorporating linguistic priors specific to Arabic morphology and syntax.

REFERENCES

[1] E. B. Page, “The imminence of... grading essays by computer,” *The Phi Delta Kappan*, vol. 47, no. 5, pp. 238–243, 1966.

[2] Z. Ke and V. Ng, “Automated essay scoring: A survey of the state of the art.,” in *IJCAI*, vol. 19, 2019, pp. 6300–6308.

[3] S. Mathias and P. Bhattacharyya, “Can neural networks automatically score essay traits?” In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, J. Burstein, E. Kochmar, C. Leacock, *et al.*, Eds., Seattle, WA, USA → Online: Association for Computational Linguistics, Jul. 2020, pp. 85–91. DOI: 10.18653/v1/2020.bea-1.8. [Online]. Available: https://aclanthology.org/2020.bea-1.8.

[4] M. Chen and X. Li, “Relevance-based automated essay scoring via hierarchical recurrent model,” in *2018 International Conference on Asian Language Processing (IALP)*, IEEE, 2018, pp. 378–383.

[5] I. Persing and V. Ng, “Modeling prompt adherence in student essays,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Toutanova and H. Wu, Eds., Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1534–1543. DOI: 10.3115/v1/P14-1144. [Online]. Available: https://aclanthology.org/P14-1144.

[6] S. Mathias and P. Bhattacharyya, “ASAP++: Enriching the ASAP automated essay grading dataset with essay attribute scores,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, N. Calzolari, K. Choukri, C. Cieri, *et al.*, Eds., Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://aclanthology.org/L18-1187.

[7] F. Dong, Y. Zhang, and J. Yang, “Attention-based recurrent convolutional neural network for automatic essay scoring,” in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, R. Levy and L. Specia, Eds., Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 153–162. DOI: 10.18653/v1/K17-1017. [Online]. Available: https://aclanthology.org/K17-1017.

[8] W. Song, Z. Song, L. Liu, and R. Fu, “Hierarchical multi-task learning for organization evaluation of argumentative student essays,” in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed., Main track, International Joint Conferences on Artificial Intelligence Organization, Jul. 2020, pp. 3875–3881. DOI: 10.24963/ijcai.2020/536. [Online]. Available: https://doi.org/10.24963/ijcai.2020/536.

[9] R. Yang, J. Cao, Z. Wen, Y. Wu, and X. He, “Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 1560–1569. DOI: 10.18653/v1/2020.findings-emnlp.141. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.141.

[10] R. Kumar, S. Mathias, S. Saha, and P. Bhattacharyya, "Many hands make light work: Using essay traits to automatically score essays," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds., Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 1485–1495. DOI: 10.18653/v1/2022.naacl-main.106. [Online]. Available: https://aclanthology.org/2022.naacl-main.106.

[11] J. Xue, X. Tang, and L. Zheng, "A hierarchical bert-based transfer learning approach for multi-dimensional essay scoring," *IEEE Access*, vol. 9, pp. 125 403–125 415, 2021. DOI: 10.1109/ACCESS.2021.3110683.

[12] Z. Jiang, T. Gao, Y. Yin, *et al.*, "Improving domain generalization for prompt-aware essay scoring via disentangled representation learning," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 12 456–12 470. DOI: 10.18653/v1/2023.acl-long.696. [Online]. Available: https://aclanthology.org/2023.acl-long.696.

[13] C. Jin, B. He, K. Hui, and L. Sun, "TDNN: A two-stage deep neural network for prompt-independent automated essay scoring," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1088–1097. DOI: 10.18653/v1/P18-1100. [Online]. Available: https://aclanthology.org/P18-1100.

[14] X. Li, M. Chen, and J.-Y. Nie, "Sednn: Shared and enhanced deep neural network model for cross-prompt automated essay scoring," *Knowledge-Based Systems*, vol. 210, p. 106 491, 2020.

[15] Y. Cao, H. Jin, X. Wan, and Z. Yu, "Domain-adaptive neural automated essay scoring," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1011–1020.

[16] R. Ridley, L. He, X.-y. Dai, S. Huang, and J. Chen, "Automated cross-prompt scoring of essay traits," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 13 745–13 753.

[17] H. Do, Y. Kim, and G. G. Lee, "Prompt- and trait relation-aware cross-prompt essay trait scoring," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 1538–1551. DOI: 10.18653/v1/2023.findings-acl.98. [Online]. Available: https://aclanthology.org/2023.findings-acl.98.

[18] Y. Chen and X. Li, "PMAES: Prompt-mapping contrastive learning for cross-prompt automated essay scoring," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 1489–1503. DOI: 10.18653/v1/2023.acl-long.83. [Online]. Available: https://aclanthology.org/2023.acl-long.83.

[19] M. M. Gaheen, R. M. ElEraky, and A. A. Ewees, "Automated students arabic essay scoring using trained neural network by e-jaya optimization to support personalized system of instruction," *Education and Information Technologies*, vol. 26, pp. 1165–1181, 2021.

[20] M. M. Gaheen, R. M. ElEraky, and A. A. Ewees, "Optimized neural network-based improved multiverse optimizer algorithm for automated arabic essay scoring," *International Journal of Scientific & Technology Research*, vol. 9, pp. 238–243, 2020. [Online]. Available: `https://api.semanticscholar.org/CorpusID:229038743`.

[21] W. Alsanie, M. I. Alkanhal, M. Alhamadi, and A. O. Alqabbany, "Automatic scoring of arabic essays over three linguistic levels," *Progress in Artificial Intelligence*, pp. 1–13, 2022.

[22] A. M. Azmi, M. F. Al-Jouie, and M. Hussain, "Aaee–automated evaluation of students' essays in arabic language," *Information Processing & Management*, vol. 56, no. 5, pp. 1736–1752, 2019.

[23] A. Alqahtani and A. Al-Saif, "Automated arabic essay evaluation," in *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, 2020, pp. 181–190.

[24] R. A. Machhout and C. B. O. Zribi, "Enhanced bert approach to score arabic essay's relevance to the prompt," *Communications of the IBIMA*, vol. 2024, 2024. DOI: `10.5171/2024.176992`. [Online]. Available: `https://doi.org/10.5171/2024.176992`.

[25] R. Cummins, H. Yannakoudakis, and T. Briscoe, "Unsupervised modeling of topical relevance in l2 learner text," in *Proceedings of the 11th workshop on innovative use of NLP for building educational applications*, 2016, pp. 95–104.

[26] M. Rei and R. Cummins, "Sentence similarity measures for fine-grained estimation of topical relevance in learner essays," in *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, J. Tetreault, J. Burstein, C. Leacock, and H. Yannakoudakis, Eds., San Diego, CA: Association for Computational Linguistics, Jun. 2016, pp. 283–288. DOI: `10.18653/v1/W16-0533`. [Online]. Available: `https://aclanthology.org/W16-0533`.

[27] M. Cozma, A. Butnaru, and R. T. Ionescu, "Automated essay scoring with string kernels and word embeddings," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 503–509. DOI: `10.18653/v1/P18-2080`. [Online]. Available: `https://aclanthology.org/P18-2080`.

[28] H. Do, Y. Kim, and G. G. Lee, "Autoregressive score generation for multi-trait essay scoring," *arXiv preprint arXiv:2403.08332*, 2024.

[29] S. Li and V. Ng, "ICLE++: Modeling fine-grained traits for holistic essay scoring," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 8465–8486.

doi: 10.18653/v1/2024.naacl-long.468. [Online]. Available: https://aclanthology.org/2024.naacl-long.468.

[30] H. Do, S. Ryu, and G. G. Lee, "Autoregressive multi-trait essay scoring via reinforcement learning with scoring-aware multiple rewards," *arXiv preprint arXiv:2409.17472*, 2024.

[31] R. Ridley, L. He, X. Dai, S. Huang, and J. Chen, "Prompt agnostic essay scorer: A domain generalization approach to cross-prompt automated essay scoring," *arXiv preprint arXiv:2008.01441*, 2020.

[32] S. Li and V. Ng, "Conundrums in cross-prompt automated essay scoring: Making sense of the state of the art," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 7661–7681. doi: 10.18653/v1/2024.acl-long.414. [Online]. Available: https://aclanthology.org/2024.acl-long.414.

[33] H. Do, T. Park, S. Ryu, and G. G. Lee, "Towards prompt generalization: Grammar-aware cross-prompt automated essay scoring," *arXiv preprint arXiv:2502.08450*, 2025.

[34] H. A. Abdeljaber, "Automatic arabic short answers scoring using longest common subsequence and arabic wordnet," *IEEE Access*, vol. 9, pp. 76 433–76 445, 2021.

[35] M. Alobed, A. M. Altrad, and Z. B. A. Bakar, "A comparative analysis of euclidean, jaccard and cosine similarity measure and arabic wordnet for automated arabic essay scoring," in *2021 Fifth International Conference on Information Retrieval and Knowledge Management (CAMP)*, IEEE, 2021, pp. 70–74.

[36] R. Ghazawi and E. Simpson, "Automated essay scoring in arabic: A dataset and analysis of a bert-based system," *arXiv preprint arXiv:2407.11212*, 2024.

[37] M. Alobed, A. M. Altrad, Z. B. A. Bakar, and N. Zamin, "Automated arabic essay scoring based on hybrid stemming with wordnet," *Malaysian Journal of Computer Science*, pp. 55–67, 2021.

[38] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over bert," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 39–48.

[39] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 1762–1777. doi: 10.18653/v1/2023.acl-long.99. [Online]. Available: https://aclanthology.org/2023.acl-long.99.

[40] K. Taghipour and H. T. Ng, "A neural approach to automated essay scoring," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1882–1891. doi: 10.18653/v1/D16-1193. [Online]. Available: https://aclanthology.org/D16-1193.

[41] J. Cohen, “Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit.,” *Psychological bulletin*, vol. 70, no. 4, p. 213, 1968.

[42] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.

[43] A. Doewes, N. Kurdhi, and A. Saxena, “Evaluating quadratic weighted kappa as the standard performance metric for automated essay scoring,” in *16th International Conference on Educational Data Mining, EDM 2023*, International Educational Data Mining Society (IEDMS), 2023, pp. 103–113.

[44] A. Esuli, S. Baccianella, and F. Sebastiani, “Evaluation measures for ordinal regression,” in *Intelligent Systems Design and Applications, International Conference on*, Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2009, pp. 283–287. DOI: `10.1109/ISDA.2009.230`. [Online]. Available: `https://doi.ieeecomputersociety.org/10.1109/ISDA.2009.230`.

[45] S. Albatarni, S. Eltanbouly, and T. Elsayed, “Graded relevance scoring of written essays with dense retrieval,” in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 1329–1338.

APPENDIX A: ARABIC BASELINES IMPLEMENTATION DETAILS

A.1. Feature-based

Feature-based models are implemented using Scikit-learn[1]. Table A.1 outlines the hyper-parameters we tune and their respective values. The best hyper-parameters are selected based on the highest QWK score on the development set. We extract features across four categories: surface features, syntactic features, lexical features, and N-gram features. Tables A.2 and A.3 provide a detailed breakdown of each feature along with its description.

Table A.1. Hyper-parameters for feature-based baselines.

Model	Hyper-parameter	Values
Random Forest	Max depth	[3, 4, 5, 6, 7, 8, 9, 10]
	Max features	[0.25, 0.5, 0.75, 1.0]
SVM	C	[0.1, 1, 10]
	Max iterations	[500, 1000, 1500, -1]

A.2. Regression Fine-tuning

We use the Hugging Face Transformers training pipeline to fine-tune the encoders, specifically, we use AutoModelForSequenceClassification. Where a regression head is appended to each model, and with a single output (i.e., num_labels=1), the training automatically employs Mean Squared Error loss.

For task-specific scoring, we performed a grid search over fine-tuning configurations, experimenting with freezing all layers, the last 3, or the last 6 layers, and learning rates of 2e-5, 3e-5, and 5e-5. The model was trained for 100 epochs, and the best checkpoint was selected based on the QWK metric. For cross-task scoring, we used the optimal parameters identified in task-specific scoring and trained the model for a fixed 100 epochs. The batch size was set to 16 across all experiments.

[1]`https://scikit-learn.org/`

Table A.2. List of all Extracted Features.

Feature Category	Feature Name	Description
Surface Features	Words_count	Total number of words in the text.
	Log_words_count	Logarithm of the total number of words.
	Unique_words_count	Number of distinct words in the text.
	Log_unique_words_count	Logarithm of the number of unique words.
	Average_word_length	Mean length of words in the text.
	Max_length_word	Length of the longest word.
	Min_length_word	Length of the shortest word.
	sd_words	Standard deviation of word lengths.
	Chars_conut	Total number of characters in the text.
	Hmpz_count	Total number of همزة in the text
	Paragraphs_count	Total number of paragraphs.
	Is_first_paragraph = 10	Whether the first paragraph has 10 or fewer words.
	Average_length_paragraph	Mean length of paragraphs.
	Max_length_paragraph	Length of the longest paragraph.
	Min_length_paragraph	Length of the shortest paragraph.
	Has_parentheses	Indicates whether the text contains parentheses.
	Has_colon	Indicates whether the text contains a colon.
	Has_question_mark	Indicates whether the text contains a question mark.
	Sentences_count	Total number of sentences.
	Average_length_sentence	Mean length of sentences.
	Max_length_sentence	Length of the longest sentence.
	Min_length_sentence	Length of the shortest sentence.
	sd_sentence	Standard deviation of sentence lengths.
Syntactic Features	noun_count	Total number of nouns in the text.
	verb_count	Total number of verbs in the text.
	adj_count	Total number of adjectives in the text.
	punc_count	Total number of punctuation marks in the text.
	pron_count	Total number of pronouns in the text.
	pre_count	Total number of prepositions in the text.
	adv_count	Total number of adverbs in the text.
	conj_count	Total number of conjunctions in the text.
	num_count	Total number of numerical values in the text.
	misspelled_count	Total number of misspelled words in the text.
	inna_count	Total number of إن وأخواتها in the text
	kana_count	Total number of كان وأخواتها in the text

Table A.3. List of all Extracted Features.

Feature Category	**Feature Name**	**Description**
Lexical Features	stop_words_count	Total number of stop words in the text.
	words_without_stopwords	Total number of words excluding stop words.
	first_paragraph_intro_words	Whether first paragraph contains introductory words.
	last_paragraph_conc_words	Whether last paragraph contains concluding words.
	lexical density	Ratio of content words (N, V, ADJ, ADV) to total words number.
N-gram Features	unigram and bigram	Single words frequency (unigrams) and 2 consecutive words (bigrams).