THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

COMBINING AUTOMATED AND HUMAN CLASSIFICATION DECISIONS

SURAJ MATAM KUMAR
Summer 2025

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Rebecca Passonneau
Professor of Computer Science and Engineering
Thesis Supervisor

Sencun Zhu
Associate Professor of Computer Science and Engineering
Thesis Honors Adviser

* Electronic approvals are on file.

ABSTRACT

This thesis investigates how relational neural models and selective deferral strategies can improve the reliability of automated short-answer grading (ASAG). While most ASAG systems focus on overall accuracy, this work looks more closely at how models behave on borderline responses, how their errors can be interpreted, and how they might work alongside human graders. Using the I-STUDIO dataset, we compare two graph-based models—SFRN and AsRRN—on both accuracy and patterns of disagreement. We use selective prediction with a logistic regression confidence model and apply a joint training strategy (JTSP) that helps the system decide when to pass questions to a human. Custom scripts analyze misclassified responses by type, confidence level, and language features. Results show that both models achieve comparable overall accuracy (~82%) but diverge systematically on low-confidence inputs. The JTSP framework improves deferral accuracy while limiting the number of human-labeled cases, achieving a deferral rate under 3% with accuracy gains above 2%. Additionally, we test cross-domain generalization by applying the learned deferral policy to the MNLI dataset. Overall, combining model confidence with learned deferral seems to make grading more reliable and better suited for shared decision-making between models and humans.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to extend a huge thanks to my thesis supervisor, Dr. Rebecca Passonneau, for her incredible support and mentorship over the past two years. I am especially grateful for the opportunity she provided me to work alongside her on this project, the insightful and educational weekly meetings, and the patient guidance that has taught me so much about research and natural language processing. I also owe a special thanks to Zhaohui Li, who was willing to help me run experiments and answer any questions I had. Without these two, the papers that this thesis builds off would not be possible. Thanks as well to my honors advisor, Dr. Sencun Zhu, for his administrative support along the way. Finally, to my friends and family, I want to thank you for your constant encouragement and support that kept me grounded during this process.

Chapter 1

INTRODUCTION

Automated classification —assigning predefined labels to data —has become increasingly common in various domains from identifying harmful online content to diagnosing medical conditions and assessing students' written responses to open-ended questions. In educational settings, automated assessment of students' written responses offers significant benefits, including faster feedback and reduced workload for instructors. However, relying solely on automated assessment introduces important risks. Automated systems can make errors, particularly when interpreting nuanced or ambiguous student answers, potentially leading to inaccurate assessments that negatively affect students. Because formative feedback influences learning trajectories, even modest misclassifications can mislead both students and instructors. To address these risks, recent attention has turned toward strategies for effective human-AI collaboration, combining the more nuanced judgment capabilities of human graders with the efficiency of automated systems. This thesis investigates how an automated system can decide, for each student answer, whether to trust its own prediction or defer to human review.

To successfully integrate automated grading systems into human-centered workflows, several key concepts must be clarified. Deferral (also known as selective prediction) refers to a model's ability to abstain from making predictions on uncertain or borderline cases, instead routing these cases back to human reviewers. This strategy reduces the chance of critical errors but requires clear criteria for when to defer, such as setting thresholds on model confidence. Model confidence, often expressed mathematically as softmax probability, refers to the

automated system's internal measure of certainty regarding a prediction, whereas human trust captures a grader's willingness to rely on these predictions. Building and maintaining this trust critically depends on transparency. While full explanatory transparency is desirable, this thesis focuses on the decision-level notion of when to trust or defer rather than on generating explicit explanations.

Recent advances in relational classification models, particularly the Semantic Feature-wise Transformation Relation Network (SFRN) and the Answer-state Recurrent Relational Network (AsRRN), have significantly improved accuracy in formative assessment of student responses (Li, Tomar, & Passonneau, 2021; Li, Lloyd, Beckman, & Passonneau, 2023b). Although these models frequently achieve comparable overall accuracy, preliminary analysis indicates they differ systematically in the types of errors they commit (see Chapter 3). Selective prediction strategies, such as Joint Training for Selective Prediction (JTSP), complement these models by incorporating deferral policies into training (Li & Passonneau, 2024). JTSP learns when to defer to human judgment by leveraging calibrated confidence scores produced by relational encoders, enabling explicit and observable deferral boundaries that support human–AI collaboration.

This thesis addresses a central question: When should an automated short-answer assessment model rely on its own prediction, and when should it defer to a human? To explore this, the thesis examines three core components:

1. It compares two relational models —SFRN and AsRRN—on the I-STUDIO dataset of 653 short, open-ended student responses (Penn State Data Commons), showing that although both models achieve similar overall accuracy, they differ in the specific kinds of mistakes they make (Li, Lloyd, Beckman, & Passonneau, 2023b).

2. It evaluates selective prediction strategies that allow a model to defer uncertain cases to human reviewers. In particular, it tests JTSP, a method that trains a classifier alongside a deferral policy to decide when deferring improves outcomes (Li & Passonneau, 2024).
3. It analyzes conditions under which each model is more or less reliable, identifying scenarios in which deferral is likely to be beneficial, while also acknowledging the limitations of imperfect human labels.

Experiments focus on the I-STUDIO dataset, which contains short, open-ended responses from introductory statistics courses labeled as correct, partially correct, or incorrect, divided into training, validation, and test subsets. To assess applicability beyond educational contexts, JTSP was also trained on a distinct natural language task: entailment classification using the Multi-Genre Natural Language Inference (MNLI) dataset (Williams et al., 2018; Hugging Face, 2021).

The rest of this thesis proceeds as follows: Chapter 2 introduces the classification models and their architectures. Chapter 3 analyzes how the models differ in their error patterns. Chapter 4 presents and evaluates the deferral strategies, including both logistic regression and JTSP.

Chapter 2

DESCRIPTION OF TWO RELATIONAL MODELS (SFRN AND ASRRN)

2.1 The ASAG Problem and Relational Models

2.1.1 The ASAG Problem

Automated Short Answer Grading (ASAG) refers to the task of evaluating student responses to open-ended questions that typically require short, free-text answers. Unlike multiple-choice assessments, short answer questions encourage students to demonstrate conceptual understanding and reasoning. However, manually grading these responses is time-consuming, inconsistent, and often subject to human bias. The goal of ASAG is to develop computational models that can automatically (and reliably) assign correctness scores to student answers.

2.1.2 Why Relational Models?

Although many deep learning approaches have been applied to ASAG, standard architecture often reduces the problem to encoding and concatenating text sequences (e.g., question plus student answer) before classification. This strategy may overlook nuanced relationships among multiple text sources: the question prompt itself, the reference solution(s), and the student's free-text answer

Relational models aim to represent these interdependencies explicitly. Rather than treating the combined text as a single unit, they model the roles and interactions among

individual components. For instance, a student's response may align partially with one reference answer and fully with another; relational architectures are designed to capture such distinctions. These interactions are typically encoded as learned relational vectors or transformation functions. This approach supports more structured reasoning over the input texts and improves the model's ability to assess correctness in a principled way.

This chapter describes two relational models designed for ASAG: SFRN and AsRRN.

2.2 The Semantic Feature-Wise Transformation Relation Network (SFRN)

2.2.1 Relational Triple Encoding

SFRN (Li et al., 2021) introduces a method for modeling interactions among three core text components in ASAG: the question (Q), reference answer(s) (R), and student response (A). Standard classification architectures often encode these multiple texts (e.g., question plus student answer) separately and then concatenate them into a single flattened vector (see, e.g., Devlin et al., 2019 for a typical transformer-based encoding). However, they generally do not introduce an explicit "relation vector" that models how these text sources interact as a structured triple.

By contrast, SFRN ensures that each text component remains distinct at the encoding stage, which is crucial for constructing a relational representation in the next step. In practice, SFRN can encode each Q, R, and A either with a baseline LSTM classifier network or with a pretrained language model such as BERT (Devlin et al., 2019). These encoding methods convert the raw text of each component into dense semantic embeddings suitable for relational modeling.

2.2.2 Learning Relation Vectors

Once embeddings for Q, R, and A are obtained, SFRN constructs relational vectors to capture their joint interactions. For each QRA triple—consisting of one encoded question vector q, one reference-answer vector rj, and one student-answer vector a—a learned function gθ processes their concatenation [q, rj, a]. Rather than relying on the raw concatenated input, this function (implemented as a multilayer perceptron) projects the combined vector into a learned representation space. The result is a relational vector lj that captures the semantic relationship among the three components.

When multiple reference answers exist for a single question, this process results in multiple relational vectors —each independently representing the distinct semantic relationship between the student response (and its associated question) and an individual reference answer.

2.2.3 Feature-Wise Transformation Fusion

An innovation of SFRN is the Semantic Feature-wise Transformation (SFT), which integrates the set of relational vectors $\{l_j\}$ into a single representation. Unlike fixed pooling methods (e.g., summation or averaging), SFT applies feature-wise scaling and shifting based on the context of each QRA triple. Specifically, for each relational vector lj, the model constructs a conditioning vector $c_j = [q. a, r_j]$, which provides contextual information about the specific QRA triple. The SFT module uses this context to guide fusion through learnable functions α and β, implemented as multilayer perceptrons (MLPs):

$$\mathrm{SFT}(C, L, n) = \sum_{j=1}^{n} \left(\alpha(c_j) \odot l_j + \beta(c_j) \right)$$

where $\odot$ denotes element-wise multiplication. The output is a single fused relational vector that reflects an optimized integration of all reference-answer interactions. This vector is then passed through a final MLP classifier trained using a cross-entropy loss to predict the correctness label of the student's response.

As Figure 1 shows, the SFT layer integrates multiple Q–R–A relation vectors before classification.

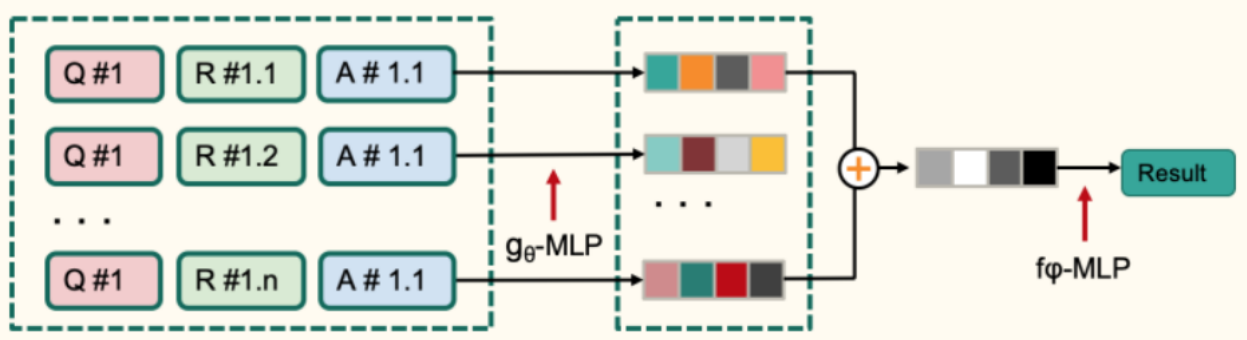

Figure 1. Semantic Feature-wise Transformation Relation Network architecture. From Li et al. (2021).

2.3 Answer-State Recurrent Relational Network (AsRRN)

2.3.1 Graph-Based Architecture

AsRRN (Li et al., 2023b) introduces a graph-based architecture for ASAG, with a special emphasis on constructed response questions prevalent in STEM education. Rather than modeling

QRA triples independently, AsRRN builds a dependency graph in which each node corresponds to a distinct input type: context (C), question (Q), reference answers (R), and student answer (A). The edges between nodes capture directed semantic dependencies and are implemented as learned relational functions.

Each node is encoded using pretrained language models such as BERT, and directed edges define the information flow: C → Q → R → A. A central super-node (S) is also included and fully connected to all other nodes. This super-node aggregates global context, ensuring coherence across the graph.

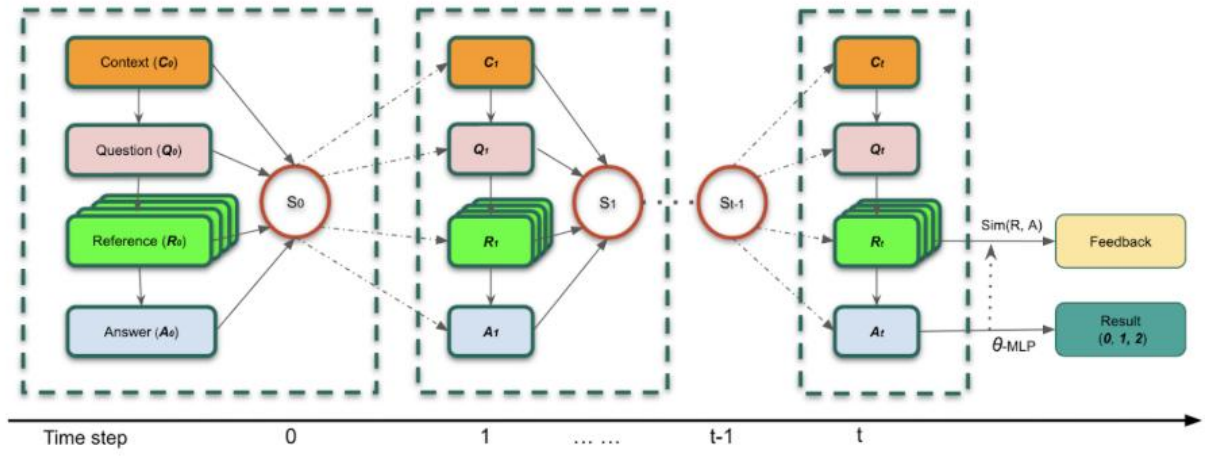

Figure 2. Answer-State Recurrent Relational Network architecture.

From Li et al. (2023b).

As shown in Figure 2, this graph structure enables richer interaction modeling than simpler or non-relational alternatives.

2.3.2 Recurrent Message Passing

A defining feature of AsRRN is its use of recurrent message passing across the graph, differentiating it significantly from single-pass models like SFRN. This method uses iterative propagation of relational information across the graph, allowing for deeper integration of inter-node information. At each time step *t*, the model computes message vectors m^t_{ij} between pairs of connected nodes *i* and *j* based on their hidden states from the previous step:

$$m_{ij}^t = f\left(h_i^{t-1},\ h_j^{t-1}\right)$$

where *f* is is a shared message function applied in parallel across the graph. For each node *i*, incoming messages from its neighbors are aggregated, and the node's hidden state is updated using another MLP *g*, which also takes the node's original input embedding *x*, as input:

$$h_i^t = g\left(h_i^{t-1},\ x_i,\ m_i^t\right)$$

This recurrence refines node representations over multiple iterations, enabling the model to integrate local and global relational information. The global supernode collects messages from all other nodes, helping to maintain a shared representation of the entire input graph. Experimental evaluations have demonstrated that this recurrent approach significantly improves inference, with optimal performance typically observed after 2 to 3 recurrent iterations.

2.3.3 Contrastive Learning for Representation

In addition to cross-entropy loss, AsRRN incorporates contrastive learning (CL) to refine its embedding space. A modified NT-Xent loss encourages embeddings of similar instances to be

close, while pushing those apart from different classes. This is adapted to reflect domain asymmetries: while “correct” responses typically align with a small set of reference answers, partially correct and incorrect responses often exhibit greater diversity.

For student answers predicted as correct, the model maximizes similarity to the average embedding of the set of correct reference answers. For answers identified as partially correct or incorrect, the model instead maximizes the similarity to individual reference examples within the corresponding class. This modification not only aligns student answer embeddings with their corresponding reference class but also compensates for the inherent diversity among partially correct responses. The contrastive loss is integrated with the standard cross-entropy loss by forming a weighted sum—allowing both losses to update the network simultaneously. This results in representations that are more distinct between correctness classes, thereby improving classification accuracy and offering potential for generating targeted automated feedback.

2.4 SFRN vs. AsRRN

2.4.1 Input Structure

SFRN and AsRRN differ fundamentally in how they organize and process input data. SFRN operates on discrete relational triples, each comprising a question, a reference answer, and a student answer. When multiple reference answers are available, each is paired with the same question and student response to form separate triples. These triples are processed independently and subsequently fused into a final relational representation. This design emphasizes localized

interactions, allowing the model to isolate and model the semantic relationship between each reference answer and the student response.

AsRRN, in contrast, adopts a graph-based structure that encodes all input components—context, question, references, and student answer—as interconnected nodes within a dependency graph. A global super-node aggregates information from all other nodes, providing a mechanism for capturing distributed semantic relationships across the full input. This architecture enables AsRRN to jointly reason about multiple input sources and their dependencies in a more holistic manner.

2.4.2 Fusion vs. Recurrence

The two models also diverge in how they aggregate relational information. SFRN relies on its SFT module to fuse multiple relation vectors into a unified representation. This process involves learned, feature-specific scaling and shifting operations that adaptively weight each Q–R–A triple based on contextual relevance. SFT is computationally efficient and well-suited to cases where relations are locally structured.

AsRRN instead employs recurrent message passing, iteratively refining node and edge representations over multiple steps. This recurrent process enhances the model's capacity to integrate relational context over the full input graph. While more computationally intensive, this iterative refinement allows AsRRN to model more complex and distributed interdependencies among the input components.

2.4.3 Complexity and Versatility

These structural differences have practical implications for computational efficiency and task suitability. SFRN's single-pass architecture offers lower computational overhead and is well suited for environments where interpretability and speed are prioritized, or where the relational structure of the input is relatively simple and well-defined.

AsRRN's recurrent, graph-based design introduces additional complexity and training time. However, this complexity enables greater representational flexibility, particularly in contexts requiring integration of heterogeneous textual inputs. In educational assessment tasks involving constructed responses, multi-part questions, or scenario-based reasoning, AsRRN's ability to capture fine-grained interrelations among components offers notable advantages.

2.4.4 Contrastive Learning

A further distinction lies in the training objectives employed. SFRN is trained solely via supervised cross-entropy loss, which encourages the model to assign high probability to the correct label while minimizing error across all classes. In contrast, AsRRN incorporates contrastive learning alongside cross-entropy. This dual-objective framework not only improves classification but also enhances the embedding space: contrastive loss pulls embeddings of semantically similar responses closer and pushes apart those from different correctness categories.

This refinement is particularly beneficial in formative assessment contexts, where subtle differences in correctness levels must be distinguished. Contrastive learning enables AsRRN to group student responses based on shared misconceptions or degrees of partial correctness,

enhancing its capacity for fine-grained assessment and targeted feedback—capabilities not directly supported by SFRN's loss function.

2.5 Transformers and the Two Relation Networks

The Transformer architecture, introduced by Vaswani et al. (2017), marked a pivotal shift in natural language processing by replacing recurrence and convolution with self-attention mechanisms. This design enables Transformers to model long-range dependencies and parallelize computations across input tokens. Given the widespread adoption of pretrained Transformer-based encoders such as BERT and GPT, it is useful to compare this paradigm to the relational modeling approach used in SFRN and AsRRN.

2.5.1 Transformer Fundamentals

The Transformer consists of two main components: an encoder and a decoder. The encoder processes token sequences using multiple layers of multi-head self-attention and feedforward networks. Each attention layer computes weighted relationships among all tokens via query, key, and value (Q/K/V) projections. Positional information is added through sinusoidal or learned embeddings to preserve word order.

The decoder, intended for generative tasks, also relies on multi-head self-attention. It masks future timesteps, explicitly preventing the model from attending to tokens that have not yet been generated and thereby avoiding any potential "cheating" by looking ahead.

Additionally, the decoder includes an encoder–decoder attention mechanism that enables it to focus on the most relevant parts of the encoder's output.

Although Transformers enable parallel processing across tokens, their self-attention mechanism requires computing multiple projections (query, key, and value) for every token and every pair of tokens. Consequently, Transformers involve significantly more parameters and higher computational costs compared to the more parameter-efficient relational networks.

2.5.2 Transformers for ASAG

In ASAG applications, Transformer-based models are typically fine-tuned for sequence classification. In this setup, the input sequence is formed by concatenating the question and the student's answer—often with special tokens marking the boundaries of each segment. The model processes this entire sequence and produces a summary representation, usually via the [CLS] token. Specifically, the [CLS] token embedding is then passed through a dense classification layer (typically with a softmax activation) to generate a probability distribution over possible correctness labels.

This architecture works well for straightforward classification tasks but encounters limitations when multiple reference answers are available or when fine-grained distinctions (e.g., partially correct vs. fully correct) are required. Some extensions attempt to address this by appending reference texts or using cross-attention mechanisms, but these often fail to model the underlying relational structure among question, references, and student answer with sufficient specificity.

2.5.3 Comparison with Relational Models

Relational models such as SFRN and AsRRN extend the Transformer's encoding capabilities by introducing explicit relational reasoning. Both models incorporate pretrained Transformers for encoding individual text components, but they depart from standard architecture by learning dedicated relation vectors (SFRN) or iterative graph-based representations (AsRRN) on top of those encodings.

This additional structure allows relational models to capture multi-way semantic dependencies that flat sequence classification architectures often overlook. Moreover, despite the computational intensity of Transformer-based attention, relational models tend to be more parameter-efficient overall. Empirical comparisons suggest that relational networks can be trained more efficiently than large-scale Transformers with additional classifier heads (Li et al., 2023c).

In summary, while Transformer-based models remain effective for general NLP tasks, relational architectures such as SFRN and AsRRN offer distinct advantages for tasks like ASAG that involve multiple interrelated inputs and require fine-grained semantic judgment.

Chapter 3

SAME ACCURACIES, DIFFERENT ERRORS: STUDENT ASSESSMENT DATA

In the previous chapter, we presented two relational models for Automated Short Answer Grading (ASAG): SFRN and AsRRN (Li, Tomar, & Passonneau, 2021; Li, Lloyd, Beckman, & Passonneau, 2023a). Each model employs distinct strategies to encode, compare, and classify student responses, yet they often reach similar accuracies when evaluated on standard metrics. Although overall accuracy provides a convenient high-level indicator of model performance, it can obscure fundamental differences in how errors arise. Understanding these differences is especially critical in educational contexts, where the cost of misclassifications can be high—for instance, when incorrect grading could lead to significant pedagogical or administrative consequences. Moreover, knowing whether two models with similar overall accuracies make the same or different errors can inform methods to combine the two models, or to select a model for human-in-the-loop approaches.

This chapter aims to investigate whether SFRN and AsRRN handle errors differently when grading the same set of student responses. We focus on various aspects of model behavior that go beyond simple accuracy, such as confusion matrices, Jaccard similarity between predicted label sets, error patterns linked to specific text properties (e.g., answer length, hedging expressions), and alignment with human annotators' majority opinions. By performing these deeper analyses, we not only gain insights into which errors each model is prone to, but also uncover important clues about when human review might be most beneficial. Although the models are comparable in aggregate performance, the findings show that they diverge in specific, repeatable ways.

3.1 Evaluation Metrics & Analytical Tools

3.1.1 Classification Metrics

Accuracy is a basic metric that reports the proportion of correctly classified student responses. If N denotes the total number of responses and k of those responses are correctly labeled, then accuracy is defined as k / N. Although accuracy provides an immediate sense of how often a model "gets it right," it does not indicate which classes are misclassified or how errors are distributed—especially in multi-class settings with fine-grained labels such as correct, partially correct, and incorrect.

To capture these nuances, we also report precision, recall, and F1-score for each class. Precision reflects how often a predicted label matches the true label (i.e., the proportion of predicted class-c instances that are truly class c), while recall captures how often the model retrieves all instances of a class. F1-score, the harmonic mean of precision and recall, balances these two measures. We include both macro-averaged and weighted F1 to summarize performance across classes: the former treats each class equally regardless of size, while the latter weights by class frequency.

3.1.2 Confusion Matrices

While aggregated metrics are useful, they gloss over class-specific nuances. The confusion matrix is a standard tool for visualizing how predictions are distributed across classes. For a multi-class problem (e.g., 0 = incorrect, 1 = partially correct, 2 = correct), a 3×3 matrix can

be formed where each row represents the true label and each column the predicted label. Diagonal cells show correct classifications; off-diagonal cells reveal how and where errors occur.

In this chapter, comparing the confusion matrices of SFRN and AsRRN is a pivotal step in highlighting which specific classes are misclassified more frequently by each model. For instance, one model might excel at recognizing partially correct answers (class 1) but occasionally overestimates correctness for clearly wrong responses (class 0), whereas the other might rarely mislabel incorrect answers as correct but often fails to promote partially correct answers. Here are the confusion matrices for both:

Table 1. Confusion matrix for SFRN on the I-STUDIO test set.

True / Pred	0	1	2
0	294	37	6
1	36	161	27
2	6	18	68

Table 2. Confusion matrix for AsRRN on the I-STUDIO test set.

True / Pred	0	1	2
0	293	41	3
1	30	178	16
2	1	20	71

These tables reveal that although both models achieve high accuracy on clearly incorrect responses (True 0, Pred 0), AsRRN produces fewer false negatives in the fully correct class (True 2, Pred 0/1) and fewer false positives in the incorrect class (True 0, Pred 2). The class-level insights motivate the deeper error-type analyses that follow in Sections 3.2 and 3.3.

3.1.3 Jaccard Similarity Scores

Although confusion matrices help us compare class-level distributions, we further quantify the overlap in predictions using Jaccard similarity. The Jaccard score between two sets A and B is defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

When applied to model outputs, A might be the set of student answers that SFRN classifies into a particular category, and B the set that AsRRN classifies similarly. The Jaccard score thus reveals how often both models agree on a given label. Low Jaccard scores for a specific class can indicate fundamentally different classification behaviors, even if both models achieve similar overall accuracies.

3.1.4 Statistical Significance Tests

To evaluate the statistical significance of performance differences, McNemar's test is applied. This test considers only the cases where the two models disagree, testing whether one consistently outperforms the other. In one analysis, chi-square tests are also used to assess

whether observed differences across categorical bins deviate from what would be expected by chance.

3.2 Script-Based Analyses

Standard metrics like accuracy, precision, and recall summarize overall performance, but they do not capture how specific textual properties affect model behavior. To delve into these details, we developed multiple Python scripts that analyze the data and classifications, each targeting a different aspect of the data or modeling assumptions.

Table 3. Overview of custom analysis scripts used in Chapter 3.

Script Name	Focus Area	Outputs & Tests
compare_confusion_matrices.py	Where SFRN and AsRRN agree or disagree in their predictions	Confusion matrix counts, Jaccard similarity by class, classification reports
length_comparison.py	Whether answer length relates to model disagreements	Accuracy by length quartile, McNemar test, basic stats
annotator_agreement_comparison.py	How model accuracy changes with annotator agreement	Accuracy, precision, recall, F1 by agreement level; McNemar test
hedging_comparison.py	Whether hedging language makes grading harder for the models	Accuracy by hedging level, normalized hedging scores, McNemar test
keyword_comparison.py	How keyword overlap affects model misclassifications	False positives by keyword match level, chi-square and McNemar tests

dataset_text_analysis.py	General language patterns in the dataset	Token counts, word lengths, vocab sizes, by-question breakdowns
qualitative_model_comparison.py	Specific examples where models agree or disagree	Annotated examples, grouped by overlap and disjoint cases

The first script (compare_confusion_matrices.py) focuses on generating and comparing confusion matrices for both models. It reads each model's category assignments (e.g., true-0, predicted-1 as "T0P1") and computes counts for every possible cell. We then evaluate agreement and disagreement across corresponding cells, measuring how often the two models classify the same answers similarly or diverge in ways that might not be apparent from overall accuracy alone. Key outputs include classification reports for each model, a consolidated spreadsheet that maps each answer to its predicted class, and a calculation of Jaccard similarity between model predictions on a per-category basis. By identifying low-Jaccard cells, we spotlight cases in which the two models fundamentally disagree, revealing distinct patterns of misclassification.

The second script (length_comparison.py) conducts a focused analysis of how student answer length correlates with divergent error behavior between AsRRN and SFRN, specifically within low Jaccard similarity cells (T0P1, T1P2, T2P0) that mark regions of high model disagreement. It reads in predicted and true labels, filters for low-agreement cases, and categorizes student answers into quartiles based on word count. For each length category, it computes accuracy separately for both models and performs McNemar's exact test to evaluate statistical significance in performance differences. Descriptive statistics (mean, median, standard deviation) for each group offer additional context. This targeted lens is justified because short or long answers may differentially impact models that fuse QRA triples (SFRN) versus those that

pass messages through a relational graph (AsRRN). By isolating responses where models disagree and stratifying by linguistic length, this script helps disentangle whether error types correlate with superficial features like verbosity or with deeper modeling differences. Its outputs support nuanced conclusions about when each model's inductive bias may become a liability.

The keyword_matching_analysis.py script looks at how much overlap there is between student answers and reference answers in terms of key statistical vocabulary, and whether that overlap affects model predictions, especially false positives. It defines a set of core stats terms (like "p-value," "confidence," "regression"), pulls up to 10 keywords from each reference answer using TF-IDF plus filtering, and then checks how many of those words show up in the student's response. It calculates both raw counts and percentage match, then groups responses by match rate to see if higher overlap correlates with more errors. For cases where the true label is not "Correct" but the model predicted it was, the script tracks how often this happens for each model and runs McNemar's test to check if there's a significant difference. It also runs chi-square tests to see if prediction patterns shift across different match-rate bins. Finally, it lists the most common keywords in misclassified answers to see what might be misleading the models.

The script, annotator_agreement_comparison.py, looks at how well SFRN and AsRRN perform across different levels of human agreement, to see whether either model is more reliable when annotators are more (or less) confident. It first calculates how many of the four annotators agreed on the label for each student response, then groups responses into "High" (4/4 agree), "Medium" (3/4), or "Low" (2 or fewer). For each group, it computes overall accuracy, precision, recall, and F1-score for both models, along with class-specific metrics for each label. It also runs McNemar's test to check if any accuracy differences between the models are statistically significant. This lets us see whether model errors tend to happen more often when humans also

disagree, or whether either model tends to break down under uncertainty. The breakdown gives a clearer picture of how model predictions relate to human confidence, which matters for deciding when a model's output should be trusted—or flagged for review.

The script hedging_language_analysis.py looks at how the presence of hedging language in student responses affects model performance. It defines a set list of hedging words (like "might," "seem," or "possibly") and calculates a normalized hedging score for each answer based on both how often these words appear and how many unique ones are used. Using these scores, responses are grouped into four categories: No Hedging, Low, Medium, and High. For each group, the script measures accuracy for AsRRN and SFRN and runs McNemar's test to see if any performance difference is statistically significant. The goal is to check whether either model struggles more when students use uncertain or cautious phrasing—something that often shows up in partially correct answers. This breakdown helps clarify whether hedging language introduces challenges for classification, and whether one model is more reliable in those cases.

Where the other scripts focus on targeted error patterns, the dataset_text_analysis.py script provides a broader textual analysis of the data. It reports answer-length distributions, the frequency of certain tokens, and differences across questions (e.g., question 2 vs. question 3). By examining word usage patterns and vocabulary sizes, we glean whether the models might be missing essential context when classifying certain question types. Furthermore, the script can zoom in on specific confusion-matrix subsets—such as T0P1 disjoint errors—for a deeper look at how language usage might relate to misclassifications.

Quantitative metrics alone cannot fully explain why models succeed or fail on specific responses. To address this, the final script (qualitative_model_comparison.py) compiles qualitative case studies by grouping examples into three categories: "overlap" (both models

correct), “AsRRN (Disjoint),” and “SFRN (Disjoint).” For each group, it records the question prompt, reference answer, and student response text in a consolidated file. These case studies reveal how model disagreements often stem from issues like borderline partial correctness, unexpected phrasing, or varying sensitivity to hedging and lexical cues. For example, SFRN may misclassify a brief but correct answer, while AsRRN might falter in cases involving ambiguity or low keyword overlap.

3.3 Findings

We begin by comparing the confusion matrices for SFRN and AsRRN to identify areas of agreement and divergence in their predictions. AsRRN slightly outperformed SFRN overall, achieving an accuracy of 83.00% compared to SFRN’s 80.09%. This advantage was most pronounced in class 2 (fully correct responses), where AsRRN attained an F1 score of 0.78 versus 0.70 for SFRN. The confusion matrices (Tables 3.1 and 3.2) reveal that AsRRN made fewer T1P2 misclassifications (16 vs. 27) and more correct T1P1 predictions (178 vs. 161). In high-frequency categories such as T0P0 (incorrect predicted as incorrect), both models performed similarly, with high Jaccard scores—0.86 for T0P0 and 0.75 for T1P1—indicating broad agreement. However, in lower-frequency or edge cases such as T1P2 (partially correct labeled as correct) and T2P0 (correct labeled as incorrect), Jaccard scores dropped below 0.20. These results suggest that while the models align on many examples, they diverge significantly in how they handle more ambiguous or challenging inputs.

We also evaluated whether differences in student response length could explain model disagreements. In a focused analysis of 102 low-Jaccard cases—those where model predictions

differed most—we found no meaningful pattern linking response length to error type. AsRRN had a higher overall accuracy on these cases (36.27% vs. 29.41% for SFRN), but the difference was not statistically significant ($\chi^2 = 30.00$, $p = 0.46$, McNemar). By quartile, AsRRN outperformed SFRN on the shortest responses (Q1: 41.38% vs. 27.59%) and in Q2, but trailed slightly in Q3. In Q4, which includes the longest responses (up to 720 words), both models performed comparably. None of these differences reached significance (all $p > 0.46$). These findings indicate that response length is not a reliable indicator of classification difficulty for either model. As such, it would be unwise to base deferral decisions solely on answer length.

We next examined model performance across different levels of annotator agreement to test reliability under varying ground-truth confidence. In high-agreement cases (4 out of 4 annotators), AsRRN achieved 88.46% accuracy compared to SFRN's 84.62%, though the difference was not statistically significant ($p = 1.00$). Similar trends were observed in medium-agreement cases (3/4 annotators: AsRRN 88.19%, SFRN 85.19%) and in low-agreement cases ($\leq$2/4: AsRRN 70.77%, SFRN 68.21%). While AsRRN appeared slightly more stable, especially in uncertain contexts, none of the comparisons reached statistical significance. Overall, both models were less reliable when human annotators disagreed, reflecting the inherent ambiguity in these responses.

We also assessed sensitivity to hedging language—words like "might," "seems," or "possibly" that signal uncertainty. AsRRN consistently outperformed SFRN across hedging levels, with the difference reaching significance only in the "High Hedging" group ($p = 0.03$, McNemar). For non-hedged responses, AsRRN scored 87.60% versus SFRN's 82.95% ($p = 0.24$). In "Low" and "Medium" groups, the models were nearly indistinguishable. In "High Hedging," AsRRN achieved 87.04% accuracy compared to 80.25% for SFRN. This suggests that

AsRRN handles cautious or qualified responses more effectively, especially in borderline cases. Still, this advantage emerged only at the extreme end of the hedging spectrum, and is not a consistent strength across all cases.

Finally, we examined whether keyword overlap between student responses and reference answers biased model predictions, especially in generating false positives. Among 203 filtered examples where the true label was not "Correct," SFRN misclassified 21 (10.34%) as fully correct, while AsRRN misclassified 11 (5.42%). This difference approached significance (p = 0.052, McNemar) but did not cross the threshold. The average number of matched keywords in SFRN's false positives was higher (mean = 1.71) than in AsRRN's (mean = 1.18), with medians of 1 for both. Chi-square tests confirmed a strong relationship between keyword overlap and misclassification for both models—SFRN ($\chi^2 = 39.10$, $p < 0.0001$), AsRRN ($\chi^2 = 20.43$, p = 0.0004)—but the effect was weaker for AsRRN. This suggests that AsRRN places less weight on lexical surface similarity when making predictions, potentially making it more robust to misleading keyword matches.

Overall, while SFRN and AsRRN achieve similar overall accuracy, they differ in how and where they make mistakes. AsRRN is somewhat better at handling hedged language and less likely to misclassify answers based on keyword overlap, while SFRN appears more prone to over-scoring concise or keyword-heavy responses. These differences could matter in settings where certain types of misclassification carry higher risks, such as educational assessment. That said, the small sample size—especially for some error categories—means these trends should be validated on larger or more varied datasets.

3.4 Data Limitations

The I-STUDIO dataset, while valuable for initial explorations, is limited by its small size. For example, the confusion matrix analysis was based on a test set of only 653 samples distributed across three classes (with 337, 224, and 92 instances, respectively). These limited sample sizes restrict the statistical power of any evaluation and reduce confidence in observed differences in model performance. Small class sizes—especially in underrepresented categories—make it difficult to determine whether observed trends are meaningful or the result of random variation.

The small sample size also makes it harder to detect more nuanced or interaction-based error patterns. Some trends that seem meaningful here may not hold up in larger or more varied datasets. In addition, several analyses grouped responses into narrower subsets (such as high-hedging or low-Jaccard cases), which further reduced the number of examples available for comparison.

In the future, larger datasets should be employed to validate these findings more robustly. Expanding the dataset would not only improve the reliability of performance metrics and statistical tests but also allow for a more granular investigation of model behavior across diverse response types and linguistic patterns.

Chapter 4

DESCRIPTION OF WAYS TO LEARN WHEN TO DEFER TO HUMANS

4.1 Background

This chapter focuses on the challenge of deciding when an automated system should predict on its own and when it should defer to a human. As discussed in earlier chapters, models like SFRN perform well on many short answer grading tasks, but their accuracy varies depending on the question and input. In ambiguous cases, deferring to a human grader can help maintain overall reliability. Selective prediction offers a principled way to do this—using model predictions only when confidence is high, and otherwise routing uncertain cases to human reviewers.

Prior work on selective prediction typically used model confidence (e.g., softmax probabilities) as a proxy for certainty. Some approaches set a confidence threshold, while others trained a separate deferral model using features from the classifier. One such method, introduced by Li et al. (2023c), uses a logistic regression model to learn a deferral policy from features like predicted class probabilities, model confidence, and question difficulty. This policy determines, for each input, whether to accept the model's prediction or defer to a human.

Building on this idea, Li and Passonneau (2024) developed a joint training framework—Joint Training for Selective Prediction (JTSP)—which trains both the classifier and the deferral policy together. Unlike previous methods where the classifier is fixed before learning to defer, JTSP allows the classifier and deferral modules to learn from each other during training. Their results show that this leads to better selective prediction performance and improves the classifier's accuracy as well.

4.2 Learning a Logistic Regression for Deferral Decision

This section outlines a selective prediction strategy based on logistic regression, originally proposed by Li et al. (2023b). The method learns a deferral policy using features derived from SFRN outputs. The goal is to identify which instances should be passed to a human grader rather than classified automatically.

4.2.1 Overview

In this approach, the SFRN model is used to generate predictions, and rather than simply using a fixed confidence threshold like the maximum softmax probability, a logistic regression model is trained to combine several signals from this classifier's output. These signals help determine the deferral decision by reflecting both the model's confidence and aspects related to the difficulty of the question. By learning a simple linear combination of these features, the logistic regression model estimates the log odds that a particular instance should be deferred to a human grader.

4.2.2 Feature Engineering

The deferral policy incorporates several features computed from the SFRN output. These include:

- The softmax probabilities for each correctness class (P_0, P_1, P_2), which capture class-specific uncertainty.

- The maximum softmax value (Max), used as a summary measure of overall model confidence.
- A question-specific accuracy estimate (SFRN-QDiff), which accounts for the model's past performance on similar items.

Additional features, such as the average student score and inter-rater reliability (ICC) for each question, can also be included to reflect question-level grading complexity, though these are used conservatively to avoid redundancy. Together, these features form a vector input to the logistic regression model, allowing it to combine local (per-instance) and global (per-question) uncertainty cues when deciding whether to defer.

4.2.3 Model Formulation and Training

The logistic regression model computes a deferral score D_{LR} as a weighted sum of input features:

$$D_{\mathrm{LR}} = \alpha + \sum_{i} W_i F_i + \varepsilon$$

where F_i are the engineered features, W_i are the learned weights, α is a bias term, and ε represents residual error. A sigmoid function is then applied to produce a probability of deferral.

Training minimizes a weighted loss function that emphasizes correct deferral decisions. Adjusting this weight alters the trade-off between automatic predictions and deferrals. Higher penalty on incorrect non-deferrals leads the model to defer more frequently, which can improve overall system accuracy by avoiding risky predictions.

4.2.4 Observations

On the development set, the logistic regression policy outperformed simple confidence-threshold baselines. It tended to defer in cases where SFRN showed low confidence or had historically underperformed. This selective deferral behavior helped maintain grading accuracy while reducing the number of instances requiring human review.

Compared to fixed-threshold methods, the logistic regression approach offered greater flexibility by incorporating multiple features. This allowed the policy to generalize more effectively, particularly on out-of-distribution test data. It deferred more frequently on ambiguous or low-confidence cases, while relying on the classifier when predictions were more trustworthy.

Taken together, the logistic regression policy offered a simple but effective way to learn deferral decisions. It showed that even a lightweight, interpretable model could outperform heuristics by combining the right set of signals and providing a solid foundation for more complex deferral strategies, like those introduced in the next section.

4.3 Joint Training for Selective Prediction (JTSP)

This section describes the JTSP framework introduced by Li and Passonneau (2024). JTSP builds on the logistic regression approach described in the previous section by jointly optimizing the classifier and deferral policy rather than treating them as independent components. This unified training process improves both selective prediction outcomes and individual module performance by allowing the two systems to learn from each other.

4.3.1 Overview of JTSP

JTSP was developed to address a key limitation of earlier selective prediction strategies: deferral decisions are often learned only after the classifier is finalized. The central insight of JTSP is that joint training enables the classifier and deferral policy to learn complementary representations. When optimized together, the two modules better detect uncertain inputs and manage the trade-off between automated efficiency and human oversight.

4.3.2 Model Architecture

JTSP consists of two primary components: the classifier (CL) and the deferral policy (DP), each with its own encoder for generating task-specific hidden representations.

The classifier processes standard inputs—student answers, question prompts, and reference responses—through a dedicated encoder (e.g., BERT-based or SFRN) to produce hidden representations (hCL) (Devlin et al., 2019). These hidden states are then passed through a multilayer perceptron to produce a distribution over correctness labels. The classifier is trained using standard cross-entropy loss.

The DP module utilizes its own separate encoder to produce hidden representations (hDP) for the same inputs as the classifier. Unlike prior methods, the DP concatenates its own hidden representation with the classifier's [hCL, hDP]. This combined representation informs the deferral decision, allowing the DP to make more contextually informed predictions about when human judgment is necessary. This combined representation informs the deferral decision, allowing the DP to make more contextually informed predictions about when human judgment is necessary. Figure 3 illustrates the resulting two-encoder design and the three training phases.

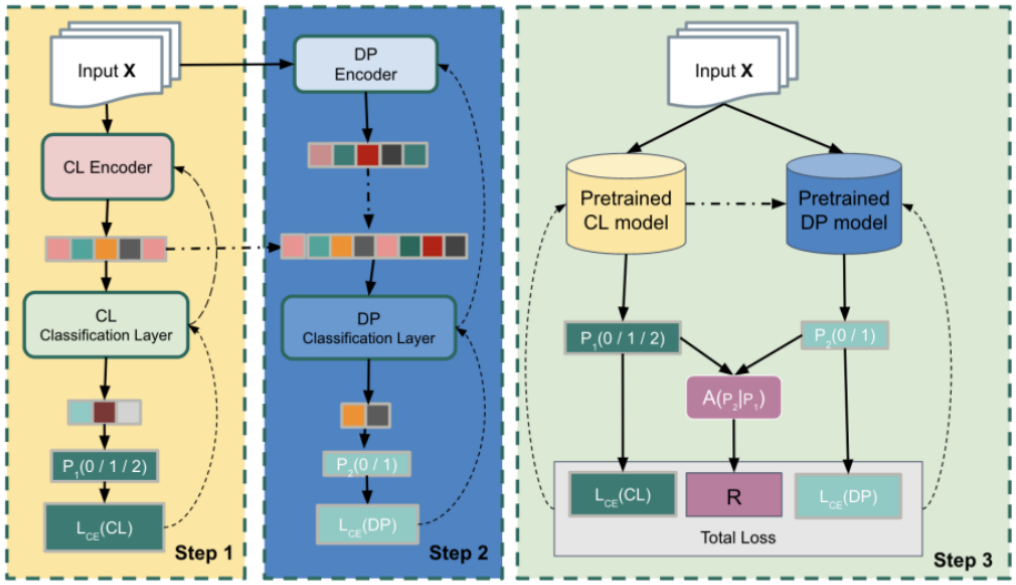

Figure 3. Joint Training for Selective Prediction architecture.

From Li & Passonneau (2024).

4.3.3 Joint Training Mechanism

To effectively coordinate training, JTSP follows structured training that begins with distinct warm-up phases and culminates in a joint optimization stage:

1. The classifier undergoes a warm-up phase independently, using standard cross-entropy loss. This establishes a baseline for the classifier.
2. CL parameters are then frozen and the deferral policy is trained in isolation, learning to make deferral decisions using the outputs of the fixed classifier.
3. After warm-up, both modules are trained simultaneously using a combined loss function that includes classifier cross-entropy loss, deferral policy cross-entropy loss, and a policy-gradient term based on a reward matrix $A = [a, b, c, d]$ which rewards desirable outcomes such as correct automated decisions and deferring incorrect ones.As Figure 4

shows, A rewards two desirable outcomes—keeping a correct prediction (a) and deferring an incorrect one (d).

	DP	
CL	-Defer	+Defer
Cor.	a	b
Incor.	c	d

$$A = [a, b, c, d] \text{ where}$$
$$\forall x \in [a, b, c, d], x \geq 0, \text{sum}([a, b, c, d]) = 1$$

Figure 4. Reward matrix used in JTSP.

High values for *a* and *d* encourage the system to keep correct automated decisions and to defer when the classifier would be wrong. From Li & Passonneau (2024).

The overall training objective is:

$$L_{\text{JTSP}} = \alpha \cdot L_{\text{ce}}(\text{CL}) + \beta \cdot L_{\text{ce}}(\text{DP}) - \gamma R$$

where L_{ce}(CL) and L_{ce}(DP) are cross-entropy losses for the classifier and deferral-policy modules, respectively. R is the expected policy reward, defined as:

$$R = \sum \pi(a \mid s) \times A(s, a)$$

where $\pi(a \mid s)$ is the probability assigned by the deferral policy to action a (defer or not), conditioned on input state s, which is usually the concatenated hidden states from both the classifier and deferral modules. The reward $A(s, a)$ is taken from the matrix in Figure 4, depending on whether the classifier was right or wrong and whether the deferral policy chose to defer. The hyperparameters α, β, and γ control how much each part of the loss contributes, with regards to classification accuracy, policy accuracy, and the reward signal. Their best values can vary quite a bit across datasets, so tuning them carefully is important.

4.3.4 Observations

The introduction of joint training in JTSP showed substantial improvements over previous selective prediction approaches. By enabling shared learning across the classifier and DP modules, JTSP significantly increased selective prediction accuracy across multiple datasets. Furthermore, the synergistic training improved individual performance metrics for both modules beyond what either could achieve independently.

However, the joint training process is sensitive to hyperparameter choices and reward configurations. Li and Passonneau (2024) reported notable variation in optimal settings across datasets, emphasizing the need for careful tuning to balance deferral rate and predictive performance.

Overall, JTSP offers a principled and effective strategy for selective prediction. By training both components in tandem, the system learns when to rely on automation and when to defer—thereby improving overall reliability in human–AI collaborative workflows.

4.4 Implementation on the MNLI Dataset

4.4.1 Overview of MNLI

The Multi-Genre Natural Language Inference (MNLI) corpus is a widely recognized benchmark for evaluating models on the task of natural language inference (NLI) (Williams, Nangia, & Bowman, 2018). Each example in MNLI consists of a pair of sentences: a premise and a hypothesis. The task is to determine the relationship between these two sentences, categorizing it as entailment, contradiction, or neutral. The dataset encompasses a diverse range

of genres, including fiction, government reports, and telephone conversations, thereby testing a model 's ability to generalize across different linguistic contexts. In total, MNLI contains approximately 433,000 annotated sentence pairs, making it one of the largest NLI datasets available (Hugging Face, 2021). For this project, we utilized the version of MNLI available through Hugging Face's datasets library, specifically the SetFit/mnli dataset. This version provides the data in a convenient format for integration with modern machine learning pipelines.

4.4.2 Differences from I-STUDIO Datasets

Adapting JTSP—originally designed for short-answer grading on the I-STUDIO dataset—to MNLI required several architectural and procedural adjustments.

First, the data format differs. I-STUDIO data are stored in CSVs with explicit question IDs, reference answers, and labels (Li et al., 2023a). In contrast, MNLI examples are accessed via Hugging Face and consist of two fields (premise, hypothesis). A custom data loader was implemented using datasets.load_dataset("SetFit/mnli"), along with a new MNLIJointDataset class to handle tokenization (Devlin et al., 2019).

Second, input structure required alignment. While I-STUDIO examples combine a question and one or more reference answers into a single string, MNLI provides two distinct text fields. To maintain compatibility with the JTSP architecture, the two MNLI fields were concatenated using [CLS] and [SEP] tokens and duplicated in a size-2 list to simulate the multi-candidate setup used in educational grading.

Another notable difference is the availability of labels. I-STUDIO's train and test sets both include ground-truth labels, facilitating straightforward training and evaluation. However,

MNLI's official test split lacks labels (with the label field set to -1) (Hugging Face, 2021). To address this, we ignored the unlabeled test split and instead carved out an internal test set from the larger training split, while also creating a validation subset from the official validation split.

Furthermore, the scale of the datasets differs markedly. I-STUDIO includes a few thousand annotated responses; MNLI contains hundreds of thousands. This introduces differences in training stability and tuning requirements.

Lastly, the tokenization strategies required adjustment. I-STUDIO employs separate tokenizers for candidate lines and for embeddings used in the deferral policy. In adapting to MNLI, we preserved this dual-tokenizer architecture, initializing a "main" tokenizer (model_name) and a "policy" tokenizer (defer_model_name) within the data module, and ensuring each received the appropriate inputs.

4.4.3 MNLI Results

Adapting JTSP to the MNLI dataset showed that the joint-training method successfully generalizes beyond automated short-answer grading. The deferral policy (DP) achieved an accuracy of 82.5%, with a DP F1 score of 0.500. The overall selective-prediction (SP) accuracy was 84.6%, closely matching the I-STUDIO JTSP SP accuracy of 85.8%. Notably, the SP F1 on MNLI slightly exceeded that of I-STUDIO, reaching 84.7% compared to I-STUDIO’s 84.3%, though the specific patterns of deferral differ across the two tasks and were not analyzed in detail here.

The deferral rate increased slightly to 3.0%, compared to I-STUDIO’s 1.84%. This modest increase likely results from MNLI's complexity, genre diversity, and longer sentence

pairs, which introduce greater ambiguity. DP accuracy was lower than on I-STUDIO (82.5% vs. 85.1%), indicating that the same reward settings produce different outcomes when applied to a multi-class entailment task with different label distributions and input formats. No additional tuning of the reward weights or training parameters was performed for MNLI beyond basic learning-rate selection.

Overall, JTSP delivered similar SP performance on MNLI as on I-STUDIO, while exhibiting modest differences in DP behavior and deferral rate. These results suggest that the joint-training framework retains much of its effectiveness in a classification context quite different from short-answer grading. The core JTSP pipeline required minimal changes beyond data loading and tokenization adjustments, and no structural modifications were needed to accommodate the shift in task.

Chapter 5

CONCLUSION

This thesis set out to answer a central question: When should a short-answer assessment model rely on its own judgment, and when should it defer to a human grader? The findings suggest that learning a selective prediction policy—especially when informed by calibrated confidence and contextual difficulty cues—offers a principled way to manage this decision. While relational models like SFRN and AsRRN perform comparably on average accuracy, their error patterns diverge in useful and predictable ways, and those patterns could potentially be exploited to support more thoughtful deferral policies.

At the model level, we observed that AsRRN's graph-based, contrastive learning approach provided better separation of the fully correct and fully incorrect classes, particularly in edge cases where SFRN tended to overpredict correctness. Meanwhile, SFRN offered a leaner architecture with more predictable degradation, showing strengths in simpler responses. The discrepancy in their confusion matrices, particularly in T2P0 and T0P2 misclassifications, points to complementary behavior rather than one model dominating outright. These divergences motivated the shift from fixed thresholds to learned deferral policies.

The logistic regression deferral policy trained on SFRN outputs and question-level metadata already outperformed naïve confidence cutoffs by reducing risky automatic predictions. Although the jointly-trained selective prediction (JTSP) framework could not be directly applied to I-STUDIO due to data size constraints, its performance on MNLI offers a proof-of-concept: deferring just 3% of inputs while still achieving a selective prediction accuracy of 84.6% shows promise for scalable, low-friction human-AI collaboration. While these results are task-specific,

the underlying mechanism—learning to predict where the model is likely to fail—generalizes beyond any single dataset.

That said, the methods come with limitations. The I-STUDIO dataset's size and class imbalance constrained the reliability of statistical comparisons and reduced power in our error analyses, especially for the partially-correct class. Script-based analyses in Chapter 3 (e.g., hedging language, annotator disagreement) yielded useful insights but remain exploratory due to low sample sizes. Additionally, deferral thresholds and confidence calibration were implemented using heuristics, leaving room for more rigorous methods, including Bayesian or distributional uncertainty estimates.

Future work should prioritize scaling up evaluation to larger, more diverse educational datasets—ideally across institutions and subject areas. This would allow better testing of how selective deferral strategies hold under varied subject domains and linguistic variation. Incorporating pedagogical cost models—for example, how harmful it is to misclassify a partially correct answer as fully wrong—could help refine deferral thresholds beyond accuracy metrics alone. From a systems perspective, integrating human-readable rationales behind "defer" decisions and testing their effects on grader trust and efficiency would push this work closer to deployment in real classroom settings. The broader challenge is not just predicting when to defer, but doing so in a way that makes sense to the human in the loop.

Ultimately, this thesis argues that effective human-AI assessment systems are not defined by flawless automation, but by strategic partnership between AI agents and humans. Knowing when not to answer—and deferring instead—is a surprisingly powerful form of intelligence. As educational technologies expand, building models that are both competent and able to partner

with humans in a somewhat transparent way will be key to designing systems that educators can actually trust.

BIBLIOGRAPHY

1. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (pp. 4171–4186). https://doi.org/10.18653/v1/N19-1423

2. Hugging Face. (2021). SetFit/mnli (Version 1.0) [Data set]. Hugging Face. https://huggingface.co/datasets/SetFit/mnli

3. Li, Z., & Passonneau, R. J. (2024). Joint training for selective prediction [Preprint]. arXiv. https://arxiv.org/abs/2410.24029

4. Li, Z., Lloyd, S., Beckman, M. D., & Passonneau, R. J. (2023a). I-STUDIO: A reliable statistical automatic short-answer assessment dataset [Data set]. Pennsylvania State University Datacommons. https://doi.org/10.26208/JFMP-V777

5. Li, Z., Lloyd, S., Beckman, M., & Passonneau, R. J. (2023b). Answer-state Recurrent Relational Network (AsRRN) for constructed-response assessment and feedback grouping. Findings of the Association for Computational Linguistics: EMNLP 2023 (pp. 3879–3891). https://doi.org/10.18653/v1/2023.findings-emnlp.254

6. Li, Z., Zhang, C., Jin, Y., Cang, X., Puntambekar, S., & Passonneau, R. J. (2023c). Learning when to defer to humans for short-answer grading. In N. Wang et al. (Eds.), Artificial Intelligence in Education: 24th International Conference, AIED 2023 (pp. 414–425). Springer. https://doi.org/10.1007/978-3-031-36272-9_34

7. Li, Z., Tomar, Y., & Passonneau, R. J. (2021). A Semantic feature-wise transformation Relation Network for automatic short-answer grading. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (pp. 6030–6040). https://doi.org/10.18653/v1/2021.emnlp-main.487

8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

9. Williams, A., Nangia, N., & Bowman, S. R. (2018). A broad-coverage challenge corpus for sentence understanding through inference. Proceedings of NAACL-HLT 2018 (pp. 1112–1122). https://doi.org/10.18653/v1/N18-1101