AMATI at BEA 2026 Shared Task 2: Automatic Short Answer Grading with Inductive Logic Programming and a Large Language Model

Alistair Willis
School of Computing and Communications
The Open University
Milton Keynes, UK
alistair.willis@open.ac.uk

Aisling Third
Knowledge Media Institute
The Open University
Milton Keynes, UK
aisling.third@open.ac.uk

Abstract

We discuss the AMATI submission to the BEA 2026 Shared Task on Rubric-based Short Answer Scoring for German. Our neuro-symbolic system uses a combination of symbolic rules, automatically learned with a form of Inductive Logic Programming, and the Mistral-large language model. We wanted to investigate whether the combination would improve overall grading performance, while using the automatically induced symbolic rules for explainability, and the LLM for robustness. We find that the combination of approached resulted in improved overall performance for the 3-way task. However, including the symbolic rules did not improve upon Mistral's performance in the 2-way test.

This paper presents our approach to the unseen answers challenges. Our team finished 6th out of 9 in the 2-way challenge, and 5th out of 8 in the 3-way challenge. In the 3-way challenge, neither our symbolic system nor the use of Mistral alone would have placed higher than 6th of the 8 competitors, illustrating the improvement of the combined approach over either of the individual approaches.

1 Introduction

Assessment is a core aspect of education. Formative assessment in particular is a valuable aid to student's self learning (Clark, 2012). Techniques such as Automatic Short Answer Grading (ASAG) ("grading" and "scoring" both appear in the literature) can potentially support scalable student learning by providing low cost feedback on low-stakes assessment (Aggarwal et al., 2025).

Recent work in ASAG has been dominated by approaches based on statistical techniques or deep learning/large language models (Lu et al., 2025). In this work, grading is typically treated as a classification task, in which student responses are assigned to one of a number of potential grades. This BEA task (Gombert et al., 2026) follows similar tasks such as the SemEval 2013 Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge (Dzikovska et al., 2013) in providing a 2 way task (student responses are classified as correct or incorrect) and a 3 way task (in which responses are classified as correct, partially correct or incorrect).

Large language models (LLMs) such as Mistral (Mistral AI, 2024) typically achieve strong empirical performance, but the resulting models are not usually interpretable by humans. Symbolic learning techniques, such as Inductive Logic Programming (ILP) (Quinlan, 1990; Muggleton and Feng, 1990) aim to induce models as logic programs rather than weighted networks. Such logic programs are generally more interpretable by humans, as well as being editable if desired. For an ILP approach to ASAG, the goal is to induce a set of rules which will assign grades from a symbolic representation of the question, rubric and student response. When applying the rules to unseen instances, the reasoning behind the allocated grade is clear. This explainability will be crucial for broader adoption of ASAG technology in the classroom.

This approach to using symbolic rules to augment neural language models is becoming of interest as the need for human-understandable explanations is increasingly recognised (Bhuyan et al., 2024; Yang et al., 2025). The neuro-symbolic system AlphaGeometry (Trinh et al., 2024) has similarly been developed to generate human readable explanations of its reasoning (in this case, geometry problems). Dinu et al. (2024) present a more general framework to address the general task of embedding logic solvers into generative models.

Although theories induced by ILP are quite explanatory (Willis, 2015), LLMs have demonstrated their robustness in the face of the noisy and poorly formed language that is typical in short student responses. In this paper, we discuss our attempt to combine the explanatory nature of ILP with the

Proceedings of the 21st Workshop on Innovative Use of NLP for Building Educational Applications, pages 1217–1223

robustness of an LLM. Our system was applied to the BEA 2026 Shared Task on Rubric-based Short Answer Scoring for German (Gombert et al., 2026) for an independently verified evaluation. Our code is available through GitHub[1].

2 Grading rule induction with Answer Set Programming

ILP is an optimisation task: we want to find the logic program which most accurately classifies the training cases. As such, much of the foundational work in ILP focussed on optimising search algorithms (Muggleton and Feng, 1990; Quinlan, 1990; Muggleton, 1995). However, some recent approaches have used Answer Set Programming (ASP) solvers to handle the optimisation (Law et al., 2014; Cropper et al., 2022).

We follow the same strategy, using the ASP solver *clingo* (Gebser et al., 2019) to optimise the rule coverage. This approach then implements an ILP-style rule induction: ASP is repeatedly used to optimise individual rules until a maximal set of training cases is covered.

Our approach consists of three components: a representation of student responses and rubric information as logical facts (section 2.1), a set of templates for classification rules (section 2.2), and a procedure for selecting rules that explain the training data (section 2.3).

2.1 Representing ASAG as a logic program

ILP requires that the information about the task be represented in the same language as the induced rules. As such, the first step is to represent the training data (that is, the questions, rubrics, student responses and ground truth gradings) in a logical format.

For the unseen answers task, each question is treated separately. For each task, we then have:

- a single question (identifier and text),
- a grading rubric (text for each possible grade),
- a set of student responses (each with an identifier and its own text), and
- a ground truth grade ("*score*") for each student response.

[1] https://github.com/undercertainty/AMATI_BEA_submission

We refer to each of the texts of each question, rubric and student response as a *document*. Each document was converted to a sequence of lemmas using spaCy (2020) with the `de_core_news_md` model. Each lemma is then encoded as a predicate of the form:

```
lemma_in_doc(Doc, Position, Lemma)
```

where `Doc` is the document's index, `Position` is the token index, and `Lemma` is the lemmatised form of the token. The ground truth grades are then encoded as:

```
label(ResponseID, Class).
```

where `Class` is one of {`correct`, `incorrect`} or {`correct`, `partially_correct`, `incorrect`} depending on the task.

As an example, the (slightly simplified) encoding of one of the questions from the training data is given in figure 1. In the encoding, lemmas in the question are indexed as `q`, in the rubric as `c`, and in the student responses as `r_01`, `r_02`, and so on. This encoding allows rules to refer directly to linguistic features. For example, a rule can test whether a response contains a particular concept (represented as a lemma), or whether that concept appears in the rubric but not in the question. The representation of the documents' indices in the predicates also enables rules to express relationships between documents, rather than treating the response in isolation.

2.2 Representing grading rules

Grading rules are expressed as logical clauses that specify the conditions under which a response should be assigned a particular label. Following the logic-based representation of the documents, the grading rules can be clearly expressed in the language of logic programming. For example, the `covers` predicate in figure 2 states that `rule3` applies to any student response whose lemmatised form contains the lemmas `kurve` and `steigen`. The additional clause `I < J` also allows us to express that for a match, the lemmas must appear in that order (ie. `kurve` precedes `steigen`). The `predicted_grade` states that any response which matches the condition in `covers` is graded as `partially_correct` by the system.

A more complex example is given in figure 3. Here, a response is covered by the rule if the response contains the lemma `reagieren`, the rubric

```
lemma_in_doc(q, 0, "welche").
lemma_in_doc(q, 1, "dir").
lemma_in_doc(q, 2, "bereits").

      ⋮

lemma_in_doc(c, 0, "die").
lemma_in_doc(c, 1, "schüler").
lemma_in_doc(c, 2, "innen").

      ⋮

lemma_in_doc(r_01, 0, "die").
lemma_in_doc(r_01, 1, "steigung").
lemma_in_doc(r_01, 2, "der").

      ⋮

lemma_in_doc(r_02, 0, "die").
lemma_in_doc(r_02, 1, "tangente").
lemma_in_doc(r_02, 2, "berührt").

      ⋮

grade(r_01, correct).
grade(r_02, incorrect).
```

Figure 1: Encoding of training data. Question id=67c1649e-b694-4813-be49-abfd65a6798e

```
covers(rule3, Response):-
  student_response(Response),
  lemma_in_doc(Response, I, "kurve"),
  lemma_in_doc(Response, J, "steigen"),
  I < J.

predicted_grade(partially_correct).
```

Figure 2: Grading rule with ordered lemmas and predicted grade

also contains `reagieren`, but `reagieren` does not appear in the question. In this case, responses which match are graded as `correct` by the system.

Figure 3 implements Pandey's (2022) concept of *key information*: new information (that is, information which does not appear in the question) which must appear in the student response. The key information can be identified as it appears in the grading rubric, but not in the question itself.

```
covers(ki_rule1, Response):-
  student_response(Response),
  lemma_in_doc(Response, _, "reagieren"),
  lemma_in_doc(c, _, "reagieren"),
  not lemma_in_doc(q, _, "reagieren").

predicted_grade(correct).
```

Figure 3: Grading rule with key information and predicted grade

2.3 Rule and theory induction via ASP

In order to generate a complete set of rules (a "theory") to grade all the responses to a given question, we use an ASP solver to generate individual rules. To generate specific rules, we define a set of generalised *rule templates*.

```
covers(ki_rule1, Response):-
  student_response(Response),
  lemma_in_doc(Response, _, Lemma),
  lemma_in_doc(c, _, Lemma),
  not lemma_in_doc(q, _, Lemma).

predicted_grade(Grade).
```

Figure 4: Rule template. `Lemma` and `Grade` are found by optimising with the answer set solver

An example template is given in figure 4. The figure represents the template form of the rule shown in figure 3. Then given the encoding of the question, rubric and a set of responses, the best *specific* rule is found from the values of:

- the choice of rule (from all the templates),
- the values of each `Lemma` variable in the chosen rule, and
- the predicted grade (ie. `Grade`)

which maximises the correct number of predicted grades from the set of student responses. To find these optimum values, we use the *clingo* answer set grounder and optimiser to find the optimum values for `Grade` and `Lemma`. (The optimised value is actually compression (Muggleton, 1995) rather than coverage).

To generate a complete theory from the initial set of student responses, we use a very simple procedure:

1. The theory is initally an empty list.
2. Find the specific rule which correctly predicts the grade for the largest number of student responses.
3. Add the specific rule to the current theory.
4. Remove all student responses which are correctly predicted by the new rule.
5. Repeat stages 2 to 4 until either all responses are covered, or no new rule can be found.

```
covers(rule1, Response):-
  student_response(Response),
  lemma_in_doc(Response, _, Lemma).

covers(rule2, Response):-
  student_response(Response),
  lemma_in_doc(Response, _, Lemma1),
  lemma_in_doc(Response, _, Lemma2).

covers(rule3, Response):-
  student_response(Response),
  lemma_in_doc(Response, I1, Lemma1),
  lemma_in_doc(Response, I2, Lemma2),
  I1 < I2.

covers(ki_rule1, Response):-
  student_response(Response),
  lemma_in_doc(Response, _, Lemma),
  lemma_in_doc(c, _, Lemma),
    not lemma_in_doc(q, _, Lemma).

covers(ki_rule2, Response):-
  student_response(Response),
  lemma_in_doc(Response, _, Lemma1),
  lemma_in_doc(c, _, Lemma1),
  lemma_in_doc(Response, _, Lemma2),
  lemma_in_doc(c, _, Lemma2),
    not lemma_in_doc(q, _, Lemma1),
    not lemma_in_doc(q, _, Lemma2).

covers(ki_rule3, Response):-
  student_response(Response),
  lemma_in_doc(Response, _, Lemma1),
  lemma_in_doc(c, _, Lemma1),
  lemma_in_doc(Response, _, Lemma2),
  lemma_in_doc(c, _, Lemma2),
    not lemma_in_doc(q, _, Lemma1).
```

Figure 5: Complete list of induction rule templates

This procedure then generates a list of rules which can be be applied to new examples. The final output of the process is a theory consisting of an ordered set of rules. Each rule defines the coverage criteria for student responses, and the predicted grade for any responses which match those criteria.

To apply the rules to new cases, each rule in the theory is applied in order. The first matching rule is used to predict the grade. If no rule matches, the prediction defaults to `incorrect`.

The performance of this system without additional input from a LLM is shown in table 1 as "Rule-based". The complete set of the rule templates is given in figure 5.

2.4 Limitations and motivation for a hybrid approach

Although some linguistic variation in the student responses is accommodated by using spaCy's lemmatisation and spell checking, the primary limitation of this method is the reliance on lexical matching. The range of possible responses from students can make the symbolic rules quite fragile. We therefore wished to explore how integrating these rules with a large language model might address this limitation.

3 Combining with a LLM

For the unseen answers task, we identified several potential approaches to combining the ILP rules with a LLM:

Mistral No ILP rules are used. For each test response, the LLM is supplied with the corresponding question and all responses in the training data for the same question, and asked to predict the correct grade.

Test ILP As *Mistral*, but also including the specific ILP prediction for the test question and the rule(s) applied to make it.

Theory As *Mistral*, but also including the complete set of rules in the theory generated by the ILP process for the task.

Test ILP and Theory As *Mistral*, including the ILP prediction and rule(s) for the test question and the complete theory.

For the unseen questions task, there is of course no "corresponding question"; our intent was to select a sample of questions from the training data instead. In practice, however, the results for the unseen questions task without the LLM did not seem to justify the time and inference cost, so we did not run the LLM cases for that task.

We decided to use the *mistral-large* model (Mistral AI, 2024) as the LLM component, via the paid Mistral API. This choice was based on its performance in the leaderboards in (Thakur, 2025) for performance in German language tasks, and its context window (the length of input it can accomodate) of 256k tokens. The prompts are in the form of templates, with placeholders to insert data, and describe the relevant scenario and what is expected from the model, and include descriptions of input and output data JSON formats. The full prompt templates are in the associated code repository. Figure 6 illustrates the prompt templates.

The output included a field for the LLM's natural language *explanation* of the given grade, to provide data for future analysis of the LLM's reasoning beyond the scope of the shared task. There are multiple avenues for this analysis, the most immediate of which is to compare LLM reasoning with the ILP rules, and evaluate its quality; further

You are a helpful and precise assistant for marking students' answers. You will be given a question, a student's answer, and the marking rubric for the question{, and a predicted score with explanation based on an inductive logic programming system}. The possible score values are "Incorrect", {"Partially Correct", }and "Correct". Your task is to evaluate each student's answer against the rubric for that question, and provide a score.

The following, between "TRAINING DATA STARTS" and "TRAINING DATA STOPS", is a JSON array of objects ...JSON details omitted ...representing the score assigned to the student's answer. Consider this as training data to improve your performance in the task.

...

{The following is the full theory generated by inductive logic programming over the training data:}

Figure 6: Prompt template (condensed for space). Scenario variations are indicated by braces.

	QWK	WP	WR	WF1
2-way				
Rule-based	0.489	0.801	0.807	0.795
Mistral	0.644	0.857	0.843	0.847
Theory	0.644	0.857	0.843	0.847
3-way				
Rule-based	0.614	0.662	0.655	0.650
Mistral	0.721	0.755	0.711	0.715
Theory	0.749	0.760	0.736	0.739

Table 1: Results for 3-way Unseen Answer tasks, showing Quadratic Weighted Kappa (QWK), and Weighted Precision (WP), Recall (WR), and F1-Score (WF1). QWK was used to rank task entrants.

possibilities include training new ILP rules from the LLM's explanations.

Testing with the provided trial data revealed that *Test ILP* (with or without the full theory) performed worse across all measures, whereas the *Mistral* and *Theory* cases performed similarly to each other, and both better than the ILP-only case.

We therefore decided to use the *Mistral* and *Theory* cases for the full evaluation.

4 Results and discussion

Our results for the 2-way and 3-way unseen answers are shown in table 1. When the ILP rules were included in the Mistral prompt, the 2-way results showed no improvement. However, for the 3-way task, including the symbolic rules in the prompt raised the overall performance (as measured by QWK) from 72.1% to 74.9%. This also represents a position in the final evaluation table: our final position was fifth of eight.

5 Conclusion

In this paper, we have discussed how the performance of a system for student response grading was improved by incorporating a set of induced symbolic rules into the neural reasoning process. Although the final results are slightly below the average system performance in the task (6^{th} of 9 in the 2-way task and 5^{th} of 8 in the 3-way task), our interest is the combination of human-interpretable rules with LLM robustness. As illustrated in section 2.2, the rules generated from the templating system are human readable (subject to being comfortable with information being presented in a formal language).

In fact, our longer term aim with this work is not (directly) to improve the overall grading performance: after all, our results show that Mistral alone gives relatively good performance. Rather, we aim to improve the explainability of awarded grades for the benefit of educators (who are ultimately responsible for grades awarded) and students (whose self-directed learning will be improved with better feedback). In a practical classroom context, it is more valuable for an AI to be able to explain why it assigned a particular grade, rather than focussing solely on performance figures.

The key question we wish to address in the next stage of this work is: Do the grades awarded by the combined system closely follow the intended interpretation of the symbolic rules? The aim of combining the symbolic rules with a LLM, is for the LLM to handle linguistic variation, while the symbolic rules handle grading of the content. As such, we want the induced rules to be editable by a human, possibly even a human who is not a computing expert, but who might be an educator in the domain of the questions. In these cases, it is vital that the application of the system primarily follows the symbolic rules (which are human editable) rather than the LLM inferences (which generally are not). Indeed, the results of LLM outputs are not always replicable (Cui and Alexander, 2026). Our further analysis will therefore concentrate on understanding the relations between the symbolic rules and the performance of the complete grading system.

Limitations

Our method makes use of a prompt to Mistral. As is always the case for such a methodology, it is always possible that alternative prompting strategies would yield different performance. In particular, we have provided English language prompts for German text. While Mistral is known to be among the most robust LLMs in handling multiple languages (Holtermann et al., 2024), prompting in the same language as the data under analysis may provide different results. Also, we restricted ourselves to a single run on each experiment.

In addition, the symbolic rules generated by the inductive logic programming process are limited to the templates provided to the system (as shown in figure 5). These rules are currently quite limited, restricted to a small number of key terms across the response, question and grading rubric. However, the initial analysis from spaCy provides extensive additional information on the parsed sentences, such as part of speech, grammatical relations, and so on. Our future work will investigate whether more information in the induced rules can be used to improve overall performance.

Ethics statement

We confirm that this work complies with the ACL code of ethics. No additional data sources were used beyond those cited in this paper, and the data provided by the team for the shared task.

Our interest in this work is driven by our belief that in issues such as grading, which can have an impact on peoples' futures, AI-based decisions must be clear and explainable. Audit trails for grading are a fundamental requirement when AI is used so that results can be challenged. We hope to be part of a movement of explainable AI behaviour in classroom environments.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments on the original draft of this paper, and particularly their suggestions for future directions for our work.

References

Dishank Aggarwal, Pritam Sil, Bhaskaran Raman, and Pushpak Bhattacharyya. 2025. "I Understand Why I Got This Grade": Automatic Short Answer Grading (ASAG) with Feedback. In *Artificial Intelligence in Education*, pages 304–318, Cham. Springer Nature Switzerland.

Bikram Pratim Bhuyan, Amar Ramdane-Cherif, Ravi Tomar, and T. P. Singh. 2024. Neuro-symbolic artificial intelligence: a survey. *Neural Computing and Applications*, 36(21):12809–12844.

Ian Clark. 2012. Formative Assessment: Assessment Is for Self-regulated Learning. *Educational Psychology Review*, 24(2):205–249.

Andrew Cropper, Sebastijan Dumančić, Richard Evans, and Stephen H. Muggleton. 2022. Inductive logic programming at 30. *Machine Learning*, 111(1):147–172. ArXiv: 2102.10556 ISBN: 0123456789.

Jiaxin Cui and Rohan Alexander. 2026. Same Prompt, Different Outcomes: Evaluating the Reproducibility of Data Analysis by LLMs. ArXiv: 2602.14349.

Marius-Constantin Dinu, Claudiu Leoveanu-Condrei, Markus Holzleitner, Werner Zellinger, and Sepp Hochreiter. 2024. SymbolicAI: A framework for logic-based approaches combining generative models and solvers. _eprint: 2402.00854.

Myroslava O. Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. 2013. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 263–274.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2019. Multi-shot ASP solving with *clingo*. *Theory and Practice of Logic Programming*, 19(1):27–82.

Sebastian Gombert, Zhifan Sun, Fabian Zehner, Jannik Lossjew, Tobias Wyrwich, Berrit Katharina Czinczel, David Bednorz, Sascha Bernholt, Knut Neumann, Ute Harms, Aiso Heinze, and Hendrik Drachsler. 2026. Report on the BEA 2026 Shared Task on Rubric-based Short Answer Scoring for German. In *Proceedings of the 21st Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2026)*. Association for Computational Linguistics.

Carolin Holtermann, Paul Röttger, Timm Dill, and Anne Lauscher. 2024. Evaluating the Elementary Multilingual Capabilities of Large Language Models with MultiQ. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4476–4494.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Mark Law, Alessandra Russo, and Krysia Broda. 2014. Inductive Learning of Answer Set Programs. In *European Conference on Logics in Artificial Intelligence (JELIA)*. ArXiv: 1608.01946.

Joan Lu, Bhavya Krishna Balasubramanian, Mike Joy, and Qiang Xu. 2025. Survey and Analysis for the Challenges in Computer Science to the Automation of Grading Systems. *ACM Computing Surveys*, 58(1).

Mistral AI. 2024. Mistral Large 3 Model Card.

Stephen Muggleton. 1995. Inverse entailment and Progol. *New Generation Computing*, 13(3-4):245–286.

Stephen Muggleton and Cao Feng. 1990. Efficient Induction of Logic Programs. *New Generation Computing*, pages 368–381. ISBN: 354063875X.

Suraj Jung Pandey. 2022. *Modelling Alignment and Key Information for Automatic Grading*. PhD Thesis, The Open University.

J R Quinlan. 1990. Learning logical definition from Relations. *Machine Learning*, 5(1990):239–266.

Ayush Thakur. 2025. Eisvogel: Evaluating German Language Proficiency.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

Alistair Willis. 2015. Using NLP to Support Scalable Assessment of Short Free Text Responses. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 243–253, Denver, Colorado. Association for Computational Linguistics.

Xiao-Wen Yang, Jie-Jing Shao, Lan-Zhe Guo, Bo-Wen Zhang, Zhi Zhou, Lin-Han Jia, Wang-Zhou Dai, and Yu-Feng Li. 2025. Neuro-Symbolic Artificial Intelligence: Towards Improving the Reasoning Abilities of Large Language Models. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10770–10778. International Joint Conferences on Artificial Intelligence Organization.