COM S 5790 Final Project: Evaluating Note-Taking Quality Using Rule-Based, BERT, and LLM Models

First Author **Second Author** **Jesus Soto Gonzalez**

Department of Computer Science
Iowa State University

first@iastate.edu, second@iastate.edu, jhsoto@iastate.edu

Abstract

This project studies how well different NLP methods can identify whether students capture important ideas while taking notes in several subjects. We use lecturer specified ideas and student-written notes to frame this as a binary classification task. Our work compares three approaches: a simple rule-based method, a BERT classifier trained on the available annotations, and a large language model evaluated with zero-shot and one-shot prompting.

1 Introduction

Note-taking is an important part of learning, but students vary in how well they record the main ideas from a lecture. Being able to automatically check whether a note contains a specific key idea could help instructors understand what students are picking up and could support research on study habits.

In this project, we work with a dataset where each lecture has a set of important ideas, called IdeaUnits, and each student provides notes divided into segments. The goal is to determine if a given IdeaUnit appears in the matching note segment. Because there is only one labeled example per topic, the task becomes a low-resource classification problem with noisy text written by students.

To explore different ways of handling this challenge, we evaluate three types of models: a rule-based approach, a BERT classifier trained on the available labels, and a large language model that relies on prompting. Our goal is to understand the strengths and weaknesses of each approach in this limited-supervision setting.

2 Related Work

3 Methodology

All methods in this project address the same prediction task: given an IdeaUnit defined by the lecturer and the corresponding segment of a student's notes, the model must decide whether the idea is present. Each team member implemented a different type of model to study how various NLP approaches handle this low-resource setting. Below we describe each model in detail.

3.1 Rule-Based Model

3.2 BERT-Based Classifier

3.3 LLM-Based Model

For the large language model approach, we used Meta's LLaMA 3 8B Instruct model (Meta AI, 2024). The implementation is organized into four main components: `preprocessing.py`, `prompts.py`, `llm_note_classifier.py`, and `evaluate_llm.py`. Together, these scripts form a simple pipeline that goes from raw CSV files to final predictions and evaluation.

Preprocessing (`preprocessing.py`). This file loads `Notes.csv`, `train.csv`, and `test.csv` and aligns them into a usable format. The key step is the `add_segment_text` function, which merges each (Topic, ID, Segment) pair with the corresponding note segment like `Segment1_Notes` and stores it in a single `segment_text` field. This gives the LLM a direct pairing of an IdeaUnit and the matching student note segment.

Prompt construction (`prompts.py`). This module defines two prompt templates, one for zero-shot and one for one-shot inference, following the in-context learning setup introduced in prior work (Brown et al., 2020). The zero-shot template includes only the task instructions, the IdeaUnit, and the student's note. The one-shot template adds a labeled example from the same topic, which contains an IdeaUnit, a note, and its correct label, so the model can see a simple demonstration of the task before answering.

Model Execution (`llm_note_classifier.py`). This script loads LLaMA 3 8B Instruct through the Hugging Face `transformers` library and runs the actual classification. For each (IdeaUnit, `segment_text`) pair, it builds a prompt using the functions in `prompts.py` and feeds it to the model. The `call_llm` function then compares the logits for the next token being "YES" or "NO" and selects the label with the higher score as the prediction.

For the one-shot condition, we use `select_one_shot_examples` to choose a single example per topic. This function computes a simple lexical overlap score between each IdeaUnit and its note and selects the positive example with the highest overlap. Using token-level lexical similarity in this way (Manning and Schütze, 2008) helps us choose cleaner and more representative examples for the prompts.

In-depth LLM Evaluation (`evaluate_llm.py`). Finally, `evaluate_llm.py` loads the saved predictions, compares them to the correct labels, and reports accuracy, precision, recall, F1 score, and a full classification report using `scikit-learn`. These metrics are used in the Results and Discussion section to compare zero-shot, one-shot, and the improved one-shot setups.

4 Experiments

4.1 Rule-Based Model

4.2 BERT-Based Classifier

4.3 LLM-Based Model

For the LLM experiments, we evaluated the model on both the small labeled training split and the full labeled test split to understand how it behaves on different amounts of data.

Early tests used free-form generation, where the model was asked to produce "YES" or "NO" directly. This approach did not work well. The model often generated full sentences such as "the note does not cover this idea," and the parser mapped these outputs to "NO." Because of this, the model predicted only the negative class and never produced a single positive prediction, so the initial results were not meaningful.

To address this problem, we switched to a probability-based decision method. Instead of generating text, we ran the full prompt through the model and checked which of the two tokens, "YES" or "NO," the model considered more likely to appear next. We selected the label with the higher probability. This approach follows common practices in prompt-based classification (Brown et al., 2020), and it removed parsing issues, made the output deterministic, and allowed the model to correctly predict both classes.

In the one-shot setup, we first used the default example provided for each topic. After that, we tested an improved example selection method. For each topic, we calculated a simple lexical overlap score between the IdeaUnit and the student's note and selected the positive example with the highest overlap. Using token-level lexical similarity in this way follows standard heuristics in text matching (Manning and Schütze, 2008) and helped choose examples that were clearer and more directly related to the idea being tested.

5 Results and Discussion

5.1 Rule-Based Model

5.2 BERT-Based Classifier

5.3 LLM-Based Model

After replacing free-form generation with next-token logit scoring, the LLM began predicting both positive and negative labels correctly, which made the evaluation meaningful.

On the small training split, zero-shot prompting reached 59.6% accuracy, slightly higher than the original one-shot setup at 58.8%. Zero-shot also showed higher precision, while one-shot showed higher recall. Because the one-shot example came from a single student's notes, it sometimes added noise rather than helping the model on this small dataset.

On the full test split, zero-shot reached 63.9% accuracy, and one-shot improved this to 64.7%. Even though the gain was small, it showed that providing one example helped the model generalize better across a much larger and more varied set of notes.

The best performance came from the improved one-shot method that selected the positive example with the highest lexical overlap between the IdeaUnit and the student note. This pushed test accuracy to 67.1% and boosted precision, suggesting that clearer and more relevant demonstrations make the prompt more effective.

Overall, the LLM performed well above the simple baseline which reaches about 51% accuracy with zero recall. These results show that the model was able to capture meaningful connections be-

tween IdeaUnits and student-written notes, and that prompt design played an important role in its performance.

A summary of all LLM results is provided in Appendix Table 1.

6 Conclusion

7 Limitations

Appendix

A LLM Results

Model	Split	Accuracy	Precision	Recall
Zero-shot	Train	0.5961	0.6235	0.4274
Zero-shot	Test	0.6392	0.5905	0.5000
One-shot	Train	0.5882	0.5785	0.5645
One-shot	Test	0.6467	0.5101	0.5628
One-shot (lex)	Train	0.6235	0.6944	0.4032
One-shot (lex)	Test	0.6713	0.5462	0.5316

Table 1: LLM-based classification results for zero-shot, one-shot, and improved one-shot prompting setups.

References

Tom B. Brown and 1 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Christopher D. Manning and Hinrich Schütze. 2008. *Foundations of Statistical Natural Language Processing*. MIT Press.

Meta AI. 2024. The llama 3 model family. https://ai.meta.com/llama/. Technical report.