Experiment: Rules vs. Learning

Building a Mood Detector with Rules and LLMs

Prof. Dr. Nicolas Meseth

In this experiment, you build a mood detector that predicts a person's mood from what they write in a chat. You start by classifying sentences by hand, then build a rule-based Python program, and finally replace the rules with a machine learning approach. The goal is to compare both approaches and reflect on their strengths and weaknesses.

Step 1: Classify by Hand

Before writing any code, work with pen and paper. You receive the following 10 sentences:

1. Not bad at all, actually.
2. Well, that could have gone worse.
3. I guess this is fine.
4. Oh great, another problem.
5. I'm not unhappy with the result.
6. Fantastic… now it's broken again.
7. I was worried, but now it seems okay.
8. That's just perfect.
9. I can live with that.
10. It's working, though I don't feel great about it.

1. Read each one carefully and label it **good**, **bad**, or **neutral**. Do it individually, and fast, in under 3 minutes.

2. Now compare your labels with a classmate. Where do you disagree? For each disagreement, write down the rule you used to justify your choice.

Class debrief: collect all rules on the board. How many unique rules emerged? Which sentences caused the most disagreement? *These are the sentences the rule-based system will fail on, you have just discovered your own test suite.*

Step 2: Project Setup

3. Make sure your Master Brick successfully connects to your computer and that you can control the LED using the Brick Viewer.

4. Create a script called `mood_rb.py`. Connect to the LED and set it to *off* initially.

5. Add a loop that continuously reads text input from the user. After the user types a message and hits ENTER, print the message back to the console. If the user enters `bye`, exit the program.

Step 3: Rule-Based Mood Detection

6. Using the rules you collected on the board in the warm-up, implement your mood detector in Python:

- Good mood -> LED green
- Bad mood -> LED red
- Cannot decide -> LED blue, neutral or unknown

What programming constructs do you need to implement your rules?

7. Test your detector on all 10 sentences from Step 1. How many does it get right compared to your own labels? Write down every case where it fails and explain why.

8. Design and add a **4th label of your own choice**, something that the three-label system cannot express. Examples: sarcastic, stressed, surprised, enthusiastic. Pick one as a group, define its keywords, and assign it a colour, for example yellow.

What does this decision teach you about the relationship between labels and the real world?

9. Test your extended detector on new edge cases: negation, sarcasm, mixed emotions, ambiguous statements. Find at least 3 sentences that fool it.

Step 4: Learning-Based Mood Detection

10. Before writing any code, open ChatGPT in your browser. Write a prompt that asks it to classify a sentence into one of your four mood labels. Try it on several sentences from Step 1. Does it get them right? Note down the exact prompt you used.

11. Now try the same prompt on the 10 sentences from Step 1 one by one. Observe how ChatGPT responds: How long is the answer? Does it always use exactly one of your four labels? Is the format consistent across responses? Write down what you notice.

12. Create a new script called `mood_ml.py` as a copy of `mood_rb.py`. Remove the rule-based detection code. Connect to the OpenAI API and send the same prompt you used in ChatGPT. Print the raw response to the console and test it on a few sentences.

What do you notice about the model's responses compared to what you expected? Is the output easy to use in your program?

13. The model's raw output is likely too long or inconsistently formatted to drive the LED directly. Adjust your prompt so that the model returns only the label, nothing else. Experiment until the output is short and predictable enough to parse reliably. What prompt changes made the biggest difference?

14. Use the label from the model's response to set the LED colour. Test the system live with the same sentences from Step 1.

15. You have now wrestled with getting consistent output from the model. OpenAI offers a feature called **structured outputs** that lets you define the exact shape of the response, the model is then forced to conform to that schema. Try it and define a response with a `label` field, restricted to your four labels, and a `reason` field, a short explanation, and update your API call to use structured output mode.

How does the response compare to what you got before? Why is this approach better when integrating an LLM into a program?

Step 5: Systematic Comparison

16. Build a test dataset in a CSV file with at least 20 sentences: include all 10 from Step 1, plus 10 new ones covering negation, sarcasm, mixed emotions, and unambiguous straightforward cases. Assign a ground-truth label to each.

17. Automate testing for both systems. Run all 20 sentences through both and record their predictions and accuracy. Print the results to the console.

18. Go beyond accuracy and visualize a **confusion matrix** for each system. Where does each system fail most? Does the learning-based system fail in the same places as the rule-based system?

19. Compare the `reason` field from the LLM across correct and incorrect predictions. Are the reasons for wrong answers still convincing? What does this tell you about the trustworthiness of an explanation?

Step 6: Reflection and Discussion

Reflect on the following questions and write down your thoughts. Discuss with your peers.

20. In Step 1, you and a classmate disagreed on some labels. If your disagreements had been used as training data for a model, what would the model have learned?

21. Where did the rule-based system work well, and where did it fail?

22. The LLM gave a reason for every decision, including the wrong ones. Does a convincing explanation make you trust the result more? Should it?

23. Which system was easier to build? Which is easier to debug? Are these the same thing?

24. Which system would you trust more in a real application, and for which *kind* of application?

25. The LLM you used was trained on billions of human-written texts, many of which were labeled or curated by humans. In what sense is it a learning-based system, just at a different scale from what you built?

26. You designed a 4th label yourself. What does the choice of labels reveal about the assumptions you are making about human emotion?