Chris Piech
CS109

Section #1
October 2nd, 2025

Section 1: Core Probability

1 Warm-Up with Music Preferences

In one prior offering of CS109, the distribution of students by year and their likelihood of liking the song *We Are the Champions* by Queen is shown in the table below. Here, "Graduate +" includes graduate students, SCPD students, coterminal students (everyone not in the other partitions). Let L_1 denote the event that a student likes the song.

Year	**% of Students**	$P(L_1 \mid \text{Year})$
Freshman	20%	0.23
Sophomore	25%	0.42
Junior	25%	0.39
Senior	10%	0.74
Graduate +	20%	0.89

What is the probability that a randomly chosen student likes *We Are the Champions* by Queen?

We are going to use the Law of Total Probability (LOTP) to solve this problem! The probability that a randomly chosen student likes this song is:

$$\begin{aligned} P(L_1) &= P(L_1 \mid \text{Freshman})P(\text{Freshman}) \\ &\quad + P(L_1 \mid \text{Sophomore})P(\text{Sophomore}) \\ &\quad + P(L_1 \mid \text{Junior})P(\text{Junior}) \\ &\quad + P(L_1 \mid \text{Senior})P(\text{Senior}) \\ &\quad + P(L_1 \mid \text{Graduate})P(\text{Graduate}) \end{aligned}$$

$$= (0.23)(0.20) + (0.42)(0.25) + (0.39)(0.25) + (0.74)(0.10) + (0.89)(0.20)$$

$$= 0.046 + 0.105 + 0.0975 + 0.074 + 0.178$$

$$= 0.5005.$$

Answer: $P(L_1) \approx 0.50$.

2 WebMD mini

In this problem, we will compute the probability that a person has a particular disease given that they present with the symptom of a fever. A person may have either *Influenza (Flu)*, *Streptococcal Pharyngitis (Strep)*, or *No Disease*. Assume these outcomes are mutually exclusive, so each person belongs to exactly one of the three groups: Flu, Strep, or No Disease.

In the general population, 8% of people have Influenza (Flu), 4.1% have Streptococcal Pharyngitis (Strep), and the remaining 87.9% have no disease. If a person has the flu, the probability they have a fever is 0.92. If they have strep, the probability they have a fever is 0.86. If they have no disease, the probability they have a fever is 0.01.

a. For each disease (Flu, Strep, No Disease), compute the probability that a person has that disease given that they present with a **fever**.

Define:

- F: the event the person has a fever.
- I: the event the person has Influenza (*Flu*).
- R: the event the person has Streptococcal Pharyngitis (*Strep*).
- N: the event the person has No Disease.

We want to compute $P(I \mid F)$, $P(R \mid F)$, and $P(N \mid F)$. By Bayes' Rule:

$$P(\text{Disease} \mid F) = \frac{P(F \mid \text{Disease})\, P(\text{Disease})}{P(F)}.$$

First, compute $P(F)$ using the Law of Total Probability:

$$P(F) = P(F \mid I)P(I) + P(F \mid R)P(R) + P(F \mid N)P(N).$$

$$P(F) = (0.92)(0.080) + (0.86)(0.041) + (0.01)(0.879) \approx 0.118.$$

Now compute each posterior:

$$P(I \mid F) = \frac{(0.92)(0.080)}{0.118} \approx 0.624.$$

$$P(R \mid F) = \frac{(0.86)(0.041)}{0.118} \approx 0.299.$$

$$P(N \mid F) = \frac{(0.01)(0.879)}{0.118} \approx 0.074.$$

Answer: Given that a person has a fever,

$$P(\text{Flu} \mid \text{Fever}) \approx 0.624, \quad P(\text{Strep} \mid \text{Fever}) \approx 0.299, \quad P(\text{No Disease} \mid \text{Fever}) \approx 0.074.$$

You can have a fever without having a disease due to hormone changes, exercise, heat stroke and stress.

b. In part (a) $P(\text{Fever}|\text{Flu}) = 0.92$ was estimated from historical data of patients who were identified as having the flu. Another option for estimating the same probability would be to use a language model with the assumption:

Assume $P(\text{Fever}|\text{Flu})$ is instead given by:

$$\texttt{prefix} = \text{“Summary of symptoms for patient with Flu: ”}$$
$$\texttt{phrase} = \text{“Has fever”}$$
$$\texttt{pr\_fever\_given\_flu} = \frac{\texttt{string\_pr(prefix + phrase)}}{\texttt{string\_pr(prefix)}}$$

What are the advantages and disadvantages of the two methods of computing $P(\text{Observation}|\text{Flu})$?

(a) Historical Data:

- Pros: Interpretable. You can see exactly how the number was calculated. It is grounded in real medical outcomes.
- Cons: May be limited by small samples. It can be hard to include/combine many rich details (e.g. “Fever of 100.2 and cough for the past 3 days”) without collecting a lot more data.

(b) Language Model `string_pr`

- Pros: Very flexible. The LLM can “reason” about detailed symptom descriptions (based on what it saw in its training data).
- Cons: Relies on a model that is not a medical professional and has known issues like overconfidence and hallucination. The number is not easy to interpret and is highly dependent on the model’s training data which is often not shared publicly.

3 Will your Friend Like This Song?

In CS109 we have 500 song recommendations! There are many randomized algorithms that can help us choose the top 16, but they rely on solving the following problem:

Imagine a student has already rated n songs as “like” or “dislike”. For any new target song i that the student has not rated yet (out of the original 500) define the event L_i as the event that the student likes song i. Estimate $P(L_i \mid \text{student's previous } n \text{ ratings})$.

To get you started, we list out a few things that you can optionally use in your solution.

Datasets
Current Quarter: Assume it is currently a few weeks into the quarter, and you have a dataset with

votes from other students in the class. Assume you have about 9 songs rated per student, and about 10 votes per song. Each student will have voted on a different randomly sampled set of 9 songs.

Historical Data: You also have a dataset from prior CS109 offerings with students and the songs that they like/dislike. Warning, the target song, and the songs they rated are likely not in the dataset.

Helper Functions

`string_pr(prompt)`
Returns the probability the LLM assigns to the *entire* input string.

`get_summary(student_song_ratings)`
Uses an LLM to produce a short natural-language description of the student's tastes.

`similarity(input_1, input_2)` Returns a number that represents how similar two inputs are in meaning (according to an LLM). For example, `similarity("rock music", "metal music")` would be a lot higher than `similarity("rock music", "broccoli")`.

This is not a problem with one correct solution. In fact, there are many possible approaches and it is still considered an open problem! The goal is to encourage you to be creative. Have fun.

These are just example solutions !! There are no "right" answers to this question.

Example 0: Baseline

You could completely ignore music preferences. You could simply assume that the probability a student will like song i is the ratio of students from the current quarter dataset that liked song i (over the total number of votes for song i). This was what most algorithms pre LLMs would do.

Example 1: Similarity (personal)

Let s_j be a song the student has already rated. For each such song, compute

`similarity(`s_j`, i)`, the similarity between s_j and the target song i.

Sort the student's rated songs by their similarity to i and select the k most similar. Let c^+ be the number of those k songs that the student liked. Then estimate:
$P(L_i|L_1, \ldots, L_n) \approx \frac{c^+}{k}$
The intuition here is that if the target song is similar to songs the student liked, the probability they will like the target song should be higher.

Example 2: Similarity (from historical data)

It is likely that in the example above, the data for the given user might be quite sparse (e.g. they have only seen 9 songs). You could instead look at the data from the historical dataset and use that as an approximation!

Let h_j be a song in the historical dataset. Loop over all songs in the historical dataset and compute `similarity(`h_j`, i)`, the similarity between a song in the historical dataset and the target song. Sort all of the songs in the historical dataset by how similar they are to the target song. Select the k songs with the highest similarity score. Of those k most similar songs, count the number of songs that are "liked by the prior CS109 students". Let's say we count a song as "liked by the prior CS109 students" if more than 50% of the students who rated it liked it. Let c^+ be the number of the k most similar songs that were liked by the prior CS109 students. Then we can approximate:

$$P(L_i|L_1, \ldots, L_n) \approx \frac{c^+}{k}$$

The intuition for this idea is if the target song is similar to songs that prior students have liked, then the probability of liking the target song should be higher!

Example 3: Bayes with Summary

Assume that you can approximate $P(\text{summary}|L_i)$ as:

$$\texttt{prefix} = \texttt{f`Here is a summary of a person who likes \{song\_i\} '}$$
$$P(\text{summary}|L_i) \approx \frac{\texttt{string\_pr(f"\{prefix\} \{summary\}")}}{\texttt{string\_pr(prefix)}}.$$

And similarly for $P(\text{summary}|L_i^C)$, if you change likes to dislikes. Then we use Bayes'

$$P(L_i|\text{summary}) = \frac{P(\text{summary}|L_i) \cdot P(L_i)}{P(\text{summary}|L_i) \cdot P(L_i) + P(\text{summary}|L_i^C) \cdot P(L_i^C)}$$

We still need to estimate the prior belief that someone would like song i. A few good options include (a) assume its 0.5 (b) estimate it from all the ratings of that song so far or (c) have the LLM help us estimate it.

Bonus Solutions These solutions are a bit beyond what you have seen so far in the course. We are putting them here for anyone curious!

Example 4: Naive Bayes

Let R_j be a random variable that represents the rating from our person to song j. The event $R_j = r_j$ is simply the event that the random variable takes on value r_j. This means that r_j will either be likes or dislikes.

Assume that you can approximate $P(R_j = r_j|L_i)$ as:

$$\texttt{rating\_j} = \texttt{`like' if rating\_j else `dislike'}$$
$$\texttt{prefix} = \texttt{f`A person who likes \{song\_i\} will \{rating\_j\} '}$$
$$P(R_j|L_i) \approx \frac{\texttt{string\_pr(f"\{prefix\} \{song\_i\}")}}{\texttt{string\_pr(prefix)}}.$$

And similarly for $P(R_j = r_j|L_i^C)$ by changing 'likes' in the prefix to 'dislikes'. If we then assume that songs are (conditionally) independent meaning that $P(\text{ratings}|L_i) = \prod_{i=1}^{n} P(R_j = r_j|L_i)$. We can then use Bayes'

$$P(L_i|\text{ratings}) = \frac{P(\text{ratings}|L_i) \cdot P(L_i)}{P(\text{ratings}|L_i) \cdot P(L_i) + P(\text{ratings}|L_i^C) \cdot P(L_i^C)}$$
$$= \frac{\left(\prod_{i=1}^{n} P(R_j = r_j|L_i)\right) \cdot P(L_i)}{\left(\prod_{i=1}^{n} P(R_j = r_j|L_i)\right) \cdot P(L_i) + \left(\prod_{i=1}^{n} P(R_j = r_j|L_i^C)\right) \cdot P(L_i^C)}$$

The assumption that ratings are conditionally independent given whether you like the song is wrong, but assumptions of this form are rather common. They are called the Naive Bayes assumption!

Example 5: Profile + String Probability

Use `get_summary` to generate a short natural-language profile of the student from their n ratings. `summary = get_summary(student_info)`. We can think of this as an approximation of all the events that the student likes and dislikes all n songs.
For a target song i we could try to estimate $P(L_i|L_1, \ldots, L_n)$ in many ways, such as:

```
prefix = 'A person with taste'
```
$P(L_i|\text{Summary}) \approx$ `string_pr(f"{prefix} {summary} will like {song_i}")`
$P(L_i^C|\text{Summary}) \approx$ `string_pr(f"{prefix} {summary} will not like {song_i}")`

⚠ But now we have a problem! The probabilities for $P(L_i|\text{Summary})$ and $P(L_i^C|\text{Summary})$ will not sum to 1. Why is that? We didn't calculate the probability of the event L_i, instead we were calculating the probability of a string (which is often really really tiny). Perhaps a reasonable assumption would be to assume that these probabilities, calculated above, are proportional to the probabilities we actually wanted. Specifically assume:

$P(L_i|\text{Summary}) = k \cdot$ `string_pr(f"{prefix} {summary} will like {song_i}")`
$P(L_i^C|\text{Summary}) = k \cdot$ `string_pr(f"{prefix} {summary} will not like {song_i}")`

You would then need to solve for the constant k that will make the probabilities sum to 1.

More Things to Consider

You could approximate $P(L_i|R_j)$. This is surprisingly difficult to do. Moreover it is really complicated to figure out how to combine the ratings for different songs.

Evaluation

How could you tell which probability estimate was the best? You could use the historical dataset, and you could run through each algorithm to predict each rating **before** showing the algorithm the result of the student rating. The one which is the best should have probabilities that match reality. For example, if you look at all the times the probability was 80%, then we would like to see 80% of the observed votes be a like!