

As you know from some of the past few chats I have been working on an app that I call chromatic chat, which is a chat app that incorporates color to highlight the emotional sentiment of a text. The stack for the project is as follows:

Frontend chat UI = Angular

RESTAPI to handle requests to backend = Fast API

Vector Database = PGVector

I have the general plan for implementing the application, but the only part that is still coming together is the logic for assigning a color to a new chat. Here is my current plan, I will need more specific help in the next few chats but for now just give your thoughts on the idea and identify any problems or issues with this approach:

Firstly, I have gotten 3 datasets for emotional sentiment tasks from Kaggle. I will be combining these to form one robust dataset. They all have a column for an example text, and a column for a label corresponding to the texts respective emotional sentiment. I want to form a label hierarchy with 3 high level sentiment classes, being positive, neutral, and negative. Neutral will be a standalone class, but both positive and negative will have distinct subclasses. These subclasses will correspond to increasingly positive and increasingly negative emotions. As a subclass becomes increasingly more positive or negative the corresponding label will also increase/decrease in value. Below is an example that highlights this idea:

Positive

Enthusiasm = 1

Happy = 2

Ecstatic = 3

Negative

Worry = -1

Dislike = -2

Hatred = -3

As the label becomes more positive/negative, its label will also increase/decrease.

The idea is to query the DB with a high k value(30-40) and sum the returned labels to get a single sentiment score value. Using the example above and a k value of 30, the most positive possible value is 90, and the most negative possible value is -90. Neutral will have a value of 0. The idea is that this range(-90 - 90), corresponds to the color frequency spectrum. Red is the lowest frequency color and Purple/Violet is the highest. So our sentiment score value maps to a color somewhere in the known color spectrum.

This is a **solid and creative core idea**, and it fits *Chromatic Chat* very well conceptually. You're essentially building a **continuous emotional signal** from discrete labeled examples and then mapping that signal to color—this is exactly the kind of abstraction that makes the app feel *intentional* rather than gimmicky.

Below are my thoughts, broken into **what works well**, **potential issues**, and **design considerations you'll want to think about soon** (not asking you to solve them yet).

What Works Well 🤞

1. Hierarchical sentiment with magnitude is a strong model

Moving from flat labels → **ordered sentiment with intensity** is the right move.

You're effectively turning sentiment into a **signed scalar**, which gives you:

- Direction (positive vs negative)
- Magnitude (how strong)

That's much more expressive than simple pos / neu / neg , and it maps *naturally* to color.

This is conceptually similar to:

- Valence scores in affective computing
- Regression-style sentiment rather than classification

So philosophically: very sound.

2. Using kNN aggregation instead of single prediction

Querying PGVector with a **high k (30–40)** and aggregating labels is a *smart* choice.

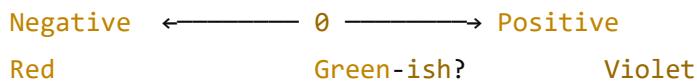
Benefits:

- Smooths noise from individual neighbors
- Reduces overconfidence
- Makes sentiment more “contextual” rather than brittle

This is especially good for **chat messages**, which are often short and ambiguous.

3. Color spectrum mapping is intuitive

Mapping:



is psychologically intuitive:

- Red = anger, danger
- Violet = excitement, energy, joy

Users won't need an explanation for why a message "feels" purple vs red.

Potential Issues / Risks

These are not deal-breakers—just things you'll want to handle deliberately.

1. Label summation assumes all neighbors are equally informative

Right now, your logic implies:

```
sentiment_score = sum(labels of top-k neighbors)
```

But:

- A very close neighbor and a barely-similar neighbor contribute equally
- Semantic similarity is ignored after retrieval

This can cause:

- Diluted sentiment
- Outliers influencing the score disproportionately

Not fatal, but something to be aware of.

2. Dataset label alignment is tricky

You're merging **multiple Kaggle datasets**, which likely have:

- Different emotion taxonomies
- Different annotation standards
- Different class distributions

Example:

- "joy" in Dataset A ≠ "joy" in Dataset B
- "neutral" may mean "emotionless" vs "mixed"

Your hierarchy *helps*, but:

- Misaligned labels will inject noise into kNN results
- Especially noticeable near the neutral boundary

This is the **biggest hidden risk** in your plan.

3. Neutral as exactly zero is fragile

Assigning:

```
neutral = 0
```

is elegant, but it assumes:

- Neutral samples dominate ambiguous inputs
- Neutral is truly emotionally flat

In practice:

- Many “neutral” texts lean slightly positive or negative
- Chat messages often contain mixed emotion

This may cause:

- Scores to collapse toward 0
 - Overuse of the same color region
-

4. Linear mapping to frequency is perceptually misleading

The electromagnetic spectrum is **not perceptually linear**.

Two equal steps in frequency:

- Do not feel equally different to the human eye

So:

- Small sentiment differences near extremes may look dramatic
- Large sentiment differences near the middle may look subtle

This isn’t wrong—but it’s a UX concern.

Design Considerations (For Later, Not Now)

Just planting seeds—you don’t need to implement these yet.

A. Emotional density vs emotional direction

Right now your system encodes:

- Direction (positive ↔ negative)
- Strength (distance from zero)

But it does *not* encode:

- Emotional complexity
- Mixed emotions

Later, you might want:

- Saturation = confidence / agreement among neighbors
- Brightness = emotional intensity
- Hue = sentiment direction

This aligns *perfectly* with color theory.

B. Edge cases: sarcasm & short messages

Messages like:

- "Great."
- "Love that for me."

Will:

- Retrieve mixed neighbors
- Produce unstable scores

Your kNN smoothing helps, but sarcasm will still slip through.

That's okay—just know it will happen.

C. Cold start bias

Early on:

- Your DB content will influence outputs heavily
- If dataset emotions dominate chat styles, colors may feel "off"

This naturally improves as chat data grows.

Overall Verdict

This is a strong, coherent approach that:

- Scales well
- Is explainable
- Maps cleanly from ML → UX
- Avoids black-box "magic sentiment numbers"

You're essentially building:

A continuous emotional embedding projected into color space

That's a *great* framing.

There are some mathematical and perceptual refinements you'll likely want to make later, but nothing here is fundamentally flawed. You're thinking about this at exactly the right level for where the project is.

When you're ready, the *next natural places to dive deeper* will be:

- How to weight neighbors
- How to normalize scores across datasets
- How to map sentiment → hue in a perceptually stable way

Whenever you want to go there, we can.

Okay thank you for the feedback, it is very helpful. The next step that I want to tackle is combining these 3 Kaggle datasets. Below will be a list of all the unique sentiment labels across the 3 datasets. I want you to sort the labels (disregarding the neutral label) into positive and negative, then create the respective subclasses for which emotions you think are more intensely positive/negative(try to stick to 3-4 subclasses):

```
['anger',
 'boredom',
 'empty',
 'enthusiasm',
 'fear',
 'fun',
 'happiness',
 'hate',
 'joy',
 'love',
 'neutral',
```

```
'relief',  
'sadness',  
'surprise',  
'worry']
```

Great, this is exactly the right next step 🤙

I'll treat this as a **semantic / affective design problem**, not a modeling one yet.

Below is a **clean, defensible hierarchy** that:

- Disregards neutral
 - Separates **positive vs negative**
 - Uses 3–4 intensity subclasses per side
 - Preserves emotional *valence* and *arousal* in a reasonable way
-

1. First: Valence Split

Positive-leaning emotions

enthusiasm
fun
happiness
joy
love
relief
surprise

Negative-leaning emotions

anger
boredom
empty
fear
hate
sadness
worry

This split is fairly uncontroversial in affective psychology.

2. Positive Sentiment Hierarchy (increasing intensity)

We want a progression from low arousal / mild positivity → high arousal / strong positivity.

Positive Subclasses

+1 : Mild / Low-Arousal Positive

Pleasant, calm, or relieving emotions

- relief
- contentment (*implicitly covered by happiness, but relief fits best here*)

Rationale:

These emotions feel good, but they're not energetic or euphoric.

+2 : Moderate / Everyday Positive

Clear positivity with moderate emotional energy

- fun
- happiness
- surprise (*when positive; it usually carries positive valence in these datasets*)

Rationale:

These are common positive chat emotions and should dominate casual conversation.

+3 : High / Energetic Positive

Strong excitement or joy

- enthusiasm
- joy

Rationale:

High arousal, outward-facing emotions—more intense than happiness.

+4 : Peak / Deep Positive

Intense, emotionally meaningful positivity

- love

Rationale:

Love is qualitatively stronger and more persistent than joy or enthusiasm.

 **Final Positive Mapping**

Label	Score
relief	+1
fun	+2
happiness	+2
surprise	+2
enthusiasm	+3
joy	+3
love	+4

3. Negative Sentiment Hierarchy (increasing intensity)

Here we want unease → sadness → fear/anger → hatred.

Negative Subclasses**-1 : Mild / Low-Arousal Negative**

Discomfort or emotional flatness

- boredom
- worry

Rationale:

Negative, but not emotionally overwhelming.

-2 : Moderate / Internalized Negative

Emotional pain or withdrawal

- sadness
- empty

Rationale:

These are heavier than boredom but still inward-facing.

-3 : High / Threat-Oriented Negative

Fear, stress, or reactive negativity

- fear
- anger

Rationale:

High arousal, strong emotional response.

-4 : Extreme / Hostile Negative

Intense rejection or hostility

- hate

Rationale:

Hate represents the strongest sustained negative emotion in your label set.

Final Negative Mapping

Label	Score
boredom	-1
worry	-1
sadness	-2

Label	Score
empty	-2
fear	-3
anger	-3
hate	-4

4. Neutral (Standalone)

Label	Score
neutral	0

This keeps your **signed scalar model clean**.

5. Why This Works Well for Chromatic Chat

- Emotional intensity increases monotonically → perfect for color gradients
- Chat-appropriate emotions dominate the middle range (-2 to +2)
- Extreme values are rare → visually meaningful when they appear
- Easy to explain and justify if you ever document the system

This hierarchy will also play very nicely with:

- kNN aggregation
- Color interpolation
- Later weighting schemes

If you want, next we can:

- Stress-test this hierarchy on example sentences
- Talk about **normalization after kNN summation**
- Or design the **exact hue mapping** (red → violet) in a perceptually sane way