Sets and basics objects for Sentiment Analysis

Let:

T = the set of tweets to be analyzed

For each tweet element of T; the text will be a sequence of tokens (words)

P = set of "Positive" (happy) words from positive_words.txt

N = set of "Negative" (angry) words from negative_words.txt

Helpers

- words(t) = normalize words in tweet *t (multiset is important because words can be repeated)*
- retweets(t) ∈ ℕ
- replies(t) ∈ ℕ

Predicates (truth-valued functions)

Predicates are functions that return True/False.

- **Pos(w) ≡ (w ∈ P)**
- **Neg(w) ≡ (w ∈ N)**
- **WordInTweet(w,t) ≡ v (w ∈ (t))** *(or "occurs at least once")*

If you want counting with repeats, define:

- **count(w, t)** ∈ **N** — number of times word **w** appears in tweet **t**. ## **3) Scores as discrete math functions**

Now define your core computed columns as functions **T** or **T** :

Positive score

If you count repeated words:

$$\text{posScore}(t) = \sum_{w \in \text{words}(t)} [\text{Pos}(w)]$$

Where [Pos(w)]** is an ****indicator**:

- **[Pos(w)] = 1** if **Pos(w)** is true
- **[Pos(w)] = 0** otherwise

Negative score

$$\text{negScore}(t) = \sum_{w \in \text{words}(t)} [\text{Neg}(w)]$$

Net score

$$\text{netScore}(t) = \text{posScore}(t) - \text{negScore}(t)$$

These definitions are exactly what you'll implement in Python.

4) Propositional logic checks (correctness rules)

These are “spec assertions” you can use to verify your program.

Column correctness

For every tweet t in T**:**

1. posScore(t) >= 0
2. negScore(t) >= 0
3. netScore(t) >= 0
4. netScore(t) = posScore(t) - negScore(t)

Polarity meaning (logical implications)

- If **(t)** > **0** then tweet is “more positive than negative”
- If **(t)** < **0** then tweet is “more negative than positive”
- If **(t)** = **0** then tie / neutral by your lexicon metric

Formally:

((t) > 0) PositiveTweet(t)

((t) < 0) NegativeTweet(t)

(Those “Tweet is positive/negative” labels are optional, but the logic is nice.)

5) CSV becomes

CSV is basically a relation/table **R** with one row per tweet:

$R \subseteq T \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{Z}$

Each row is:

$$(t, \text{retweets }(t), \text{replies }(t), \text{posScore }(t), \text{negScore }(t), \text{netScore }(t))$$

6) The graph, as a relation too

The scatterplot is the set of points:

**G = {(netScore(t),text retweets(t)) | t

So:

- **X-axis** = **netScore(t)**
- **Y-axis** = **netScore(t)**

7) Direct translation

- Turn **P** and **N** into Python **sets** for fast membership: **w in positive_words**
- Define a normalize/tokenize function = words(t)
- Compute scores using indicator logic: **score += 1 if w in P else 0**