# Self-Learning Emotion Detection System – Full Plan

## 1. Overview

This project is focused on developing a self-learning emotion detection system without the use of external libraries. The goal is to implement a keyword-based approach that understands emotional intensity, adapts over time, and improves via user feedback.

## 2. Project Structure and Key Components

Current Files: 8 (not specified, assumed to include tokenizer, normalizer, processing, etc.)

Primary Data Structures:

* - `unordered\_map<string, int>`: For emotion words and their sentiment scores
* - `unordered\_map<string, double>`: For intensifiers (adverbs) and their modifying values
* - `unordered\_map<string, int>`: For unknown word frequency tracking
* - `unordered\_map<string, pair<string, int>>`: For tracking unknown words with guessed POS and frequency
* - `unordered\_set<string>`: For verbs, adjectives, adverbs from POS files

## 3. Emotion and Intensifier System

Emotion Words: Adjectives (e.g., good, bad, happy) assigned values like +2 or -2.

Intensifiers: Adverbs (e.g., very, slightly, not) modify emotion word strength.

Formula used: Multiply intensifier values with base emotional value. Special logic for negation handling.

If intensifier < -1 or between -1 and 0, set `intensifier = -1 / intensifier` to invert the effect of negation correctly.

## 4. Input Handling Pipeline

- Normalize input (case folding, punctuation removal).

- Tokenize sentence into words.

- Pass through typo correction using edit distance for known word matching.

- Detect known emotion or intensifier using maps and sets.

- Multiply intensifier values with emotion values to get final sentiment score.

## 5. Part-of-Speech Dictionary

- Maintain external POS wordlists: adjectives, adverbs, verbs, etc.

- Use `unordered\_set` to check if a word exists in a POS category.

- When an unknown word is encountered, check context (preceding and following words) to guess POS.

- Store guessed POS along with frequency in a learning map.

## 6. Learning Unknown Words

Three approaches when encountering an unknown word:

1. 1. Ask the user if this word is an emotional word, intensifier, or irrelevant.

2. If user confirms, add to the appropriate map and save to file.

3. If marked as irrelevant, discard permanently.

4. If flagged for 'watching':

* - Increment frequency when seen in similar context.

- If frequency exceeds threshold, propose to save as learned.

- Ask for user confirmation, else move to garbage.

## 7. Feedback Loop

When the system makes a guessed decision, it provides feedback like:

- 'Did you mean this word as an emotion? [Yes/No]'

- If yes: Save immediately.

- If no: Mark as garbage and exclude from further guesses.

## 8. Avoiding Repeated Prompts

If a word is seen again but flagged for learning, it will not prompt the user repeatedly.

Instead, after frequency crosses threshold, it'll say:

-> 'I’ve remembered the word "X" as an intensifier based on repetition. Do you want to change that?'

This is a slight lie to reduce user friction but improves user experience.

## 9. Final Thoughts

The entire system is designed with flexibility and clarity in mind. It evolves with time, user interaction, and language pattern recognition. Future upgrades can include GUI, deeper NLP, and even voice-based interaction.