Assignment 3

Implementing VADER Sentiment Analysis in C

Pedram Pasandide

Due Date: 2024, 12 Nov

1 Introduction to Sentiment Analysis and VADER

Sentiment analysis is a Natural Language Processing (NLP) technique used to determine and analyze emotions, opinions, and subjective information within text data. It's widely employed to assess sentiments in social media posts, customer reviews, and other written feedback, categorizing text as positive, negative, neutral, or more granular emotional states. If you need more information, read: AWS - What is Sentiment Analysis.

VADER (Valence Aware Dictionary for sEntiment Reasoning) is a rule-based sentiment analysis tool specifically tuned for social media. It uses a predefined lexicon of words associated with specific sentiment intensities, making it effective at handling context-heavy content such as slang, emojis, and varying intensities of emotions. VADER outputs sentiment scores for positive, negative, and neutral tones, as well as a compound score that represents the overall sentiment. For more details, see: VADER Sentiment Analysis.

2 Problem Statement

In this assignment, you will implement a **simplified version** of the VADER sentiment analysis tool in C. In the future, if you are interested to develop more sophisticated VADER, you can start from here. The VADER implementation involves reading a lexicon file, identifying sentiment-bearing words within a sentence, and applying specific rules to calculate sentiment scores. You will be introduced to handling words with intensity modifiers (e.g., "very"), punctuation, ALLCAPS emphasis, and negations, simulating how VADER processes sentiment in real-world text. Inputs and Outputs are as follows:

- **Input**:
  1. A sentence string (e.g., `"VADER is very smart, handsome, and funny."`).
  2. A lexicon file (`vader_lexicon.txt`) containing words with sentiment scores and sentiment distributions from human ratings.

- **Output**: The sentiment scores for the input text, including:
  1. Positive score (pos)
  2. Negative score (neg)
  3. Neutral score (neu)
  4. Compound score (compound) - representing the overall sentiment strength, normalized to range from -1 to 1.

2.1 Lexicon File (`vader_lexicon.txt`)

The file `vader_lexicon.txt` contains words with associated sentiment values and distributions. The following table shows the format of a few lines in this file:

Table 1: The contents of given lexicon file `vader_lexicon.txt` from vaderSentiment

Word	Sentiment Value 1	Sentiment Value 2	Array of Distribution
smart	1.7	0.78102	[2, 2, 1, 2, 1, 3, 2, 0, 2, 2]
funny	1.9	0.53852	[3, 2, 2, 1, 2, 2, 1, 2, 2, 2]
handsome	2.2	0.74833	[2, 2, 2, 2, 2, 3, 4, 1, 2, 2]
guilty	-1.8	0.6	[-1, -2, -2, -3, -2, -2, -1, -2, -2, -1]

Each line contains:

- A word (word) as a string.
- Two floating-point values (Sentiment value 1 and Sentiment Value 2), representing the average sentiment score and the standard deviation, respectively.
- An array of integers (Array of Distribution) representing human sentiment ratings distribution for each word. These ratings show the range of sentiment perceptions across different people. Checkout Appendix if you need more information about distribution array.

2.2 VADER Formulation with a Numerical Example

Let's describe the formulation with an example. We want to compute the sentiment score for a given text, for example "VADER is very smart, handsome, and funny.". To calculate the VADER sentiment scores for the given sentence, VADER follows these steps:

1. **Tokenization**

   The sentence is split into words:

   `["VADER", "is", "very", "smart", "handsome", "and", "funny"]`.

2. **Identify Words in Lexicon and Ignore Non-Lexicon Words**

   VADER uses the `vader_lexicon.txt` to match words with known sentiment values:

   - `"smart"`: 1.7
   - `"handsome"`: 2.2
   - `"funny"`: 1.9
   - The words `"VADER"`, `"is"`, `"very"`, and `"and"` are not in the lexicon and are therefore ignored.

3. **Handling Others Rule in this case Intensifiers (e.g., ”very”)**

   The word `"very"` is an intensifier and slightly boosts the sentiment of the word that follows it (`"smart"`). VADER uses a **boost factor** of **0.293** for intensifiers.

   - Sentiment Score for `"smart"`:
     boosted score for `"smart"` $= 1.7 + (1.7 \times \mathbf{0.293}) = 1.7 + 0.4981 = 2.1981$
   - Sentiment Score for `"handsome"`: 2.2 (no boost)
   - Sentiment Score for `"funny"`: 1.9 (no boost)

4. **Calculate Total Sentiment and Normalize**

   Sum the adjusted sentiment scores and calculate the average:

$$total\ score = 2.1981 + 2.2 + 1.9 = 6.2981$$

$$average\ score = \frac{total\ score}{den} = \frac{6.2981}{3} = 2.09937$$

   where 3 is the total number of words that had a sentiment score.

5. **Calculate Compound Score**

   The compound score is calculated by normalizing the average sentiment score to be between -1 and 1. The formula VADER uses is:

$$compound = \frac{total\ score}{\sqrt{total\ score^2 + \alpha}}$$

   where α is normalization constant (default value is 15). For the given example:

$$compound = \frac{6.2981}{\sqrt{6.2981^2 + 15}} = 0.8518$$

6. **Calculate Positivity, Neutrality, and Negativity Scores**

   VADER categorizes each word score based on their positivity, neutrality, or negativity.

   (a) **Positive Score (pos)**
   (b) **Neutral Score (neu)**
   (c) **Negative Score (neg)**

   To simply this assignment we will skip calculating the above parameters. If are interested in the future to improve your program, you can study the Python implementation on here.

VADER returns the **final output**:

```
compound: 0.8518
```

This calculation follows VADER's scoring approach, with slight adjustments for the exact final values in the VADER library. To make sure that VADER can handle more complex sentence structures, rule like Intensifiers needs to be considered. There are **four** main rules that you **MUST** consider in your implementation as well:

1. **Intensifiers**

   Intensifiers are words that amplify or diminish the sentiment of an adjacent word. VADER applies a **boost facto**r of approximately **0.293** to increase or decrease the intensity of the sentiment for a word that follows an intensifier. Here's a list of commonly used intensifiers in daily language and on social media:

   - **Positive Amplifiers**: "absolutely," "completely," "extremely," "really," "so," "totally," "very," "particularly," "exceptionally," "incredibly," "remarkably"
   - **Negative Amplifiers**: "barely," "hardly," "scarcely," "somewhat," "mildly," "slightly," "partially," "fairly," "pretty much"

   For instance, in the sentence "That was very funny," the word "very" intensifies the positive sentiment of "funny."

2. **Punctuation Emphasis**

   VADER also uses punctuation to gauge the intensity of a sentiment. Exclamation marks in particular add emphasis, while repeated punctuation can further increase the emphasis. Rules for Punctuation are given as follows:

- **Exclamation Marks**: Each exclamation mark adds a small **boost** (usually around **0.292**) to the sentiment. For example, "funny!" would score slightly higher than "funny."
- **Repeated Exclamation Marks**: Multiple exclamation marks (e.g., "funny!!!") add progressively more emphasis up to a reasonable limit (often capped at around 3 punctuation marks in practice).
- **Question Marks**: While question marks are usually neutral, multiple question marks (e.g., "funny??") can sometimes be interpreted as uncertainty or sarcasm and may slightly reduce the sentiment. In this assignment, handling question mark is not mandatory.

For example, "funny!" would be perceived as slightly more positive than "funny," and "funny!!!" even more so.

3. **ALLCAPS Emphasis**

ALLCAPS is used to signify emphasis or shouting, which VADER interprets as heightened emotion. If a word in ALLCAPS has a sentiment value, VADER will typically increase its score by a **factor of 1.5** (for both positive and negative words). Rules for ALLCAPS:

- If at least one sentiment-laden word is in ALLCAPS, the intensity of that word is amplified.
- For example, "SMART" (in ALLCAPS) will be interpreted as a more intense positive sentiment than "smart."

So, if "smart" has a base sentiment score of 1.7, then "SMART" would be scored around $1.7 \times 1.5 = 2.55$.

4. **Negation**

Negations invert the sentiment of a word or phrase that follows them. In VADER, negations usually reduce the positive or negative sentiment by **multiplying the sentiment by -0.5**. This flips positive sentiment to negative, and negative sentiment to positive, though the overall impact is less intense than the original sentiment. Negation words commonly recognized by VADER include:

- Not: "not," "isn't," "doesn't," "wasn't," "shouldn't," "won't," "cannot," "can't"
- Negative Conjunctions: "nor," "neither"
- Other Negations: "without," "lack," "missing"

For example, "funny" has a positive sentiment score of 1.9. In "isn't funny," the presence of "isn't" before "funny" changes the score to:

$$1.9 \times -0.5 = -0.95$$

In practice, these rules for intensifiers, punctuation, ALLCAPS, and negations allow VADER to capture nuances that reflect real-world sentiment expression on social media.

3 Programming the Logic (16 points)

First, **Read the Lexicon File**. Implement a function to read the `vader_lexicon.txt` file and store each word's data in a structured format. Define `WordData` as follows:

```c
#define ARRAY_SIZE 10
#define MAX_STRING_LENGTH 5  // Adjust if needed

typedef struct {
 char word[MAX_STRING_LENGTH];
 float value1;                // Mean sentiment value
 float value2;                // Standard deviation
 int intArray[ARRAY_SIZE];    // Array of sentiment ratings
} WordData;
```

The function should dynamically allocate an array of `WordData` where each element corresponds to one word in the lexicon. This array will act as a dictionary to access each word's sentiment information.

Then **Calculate Sentiment Scores**. Define a function to analyze a given input string and calculate the **sentiment compound** score and return the value. Within this function:

- Parse the input sentence.
- Apply sentiment rules for intensifiers, punctuation, ALLCAPS, and negation as described.
- Return the final sentiment score `compound` based on the calculations shown above.

The implementation and declaration of all above functions must be in `vaderSentiment.c` and `utility.h`. **To avoid losing any points make sure your program is well structured in terms of splitting the codes into multiple files. Feel free to reach out any time**

after our lectures to discuss the design you have in your mind. Create `main.c` holding `int main()`, and test your program for the test cases mentioned in the following section.

4 Report and Makefile (4 points)

In your `Makefile.pdf`, explain the algorithms you have implemented, how to compile and run your program. Run the test cases for the following sentences and fill out the given table:

Table 2: Case Study with VADER developed in C

Sentence	Compound	
	Model in C	Python Library
VADER is smart, handsome, and funny.	0.8518	0.8316
VADER is smart, handsome, and funny!		
VADER is very smart, handsome, and funny.		
VADER is VERY SMART, handsome, and FUNNY.		
VADER is VERY SMART, handsome, and FUNNY!!!		
VADER is VERY SMART, uber handsome, and FRIGGIN FUNNY!!!		
VADER is not smart, handsome, nor funny.		
At least it isn't a horrible book.		
The plot was good, but the characters are uncompelling and the dialog is not great.		
Make sure you :) or :D today!		
Not bad at all		

IMPORTANT! To make the process of marking this assignment more efficient, please make sure the `int main()` in `main.c` will run the test and print the results for all above sentences in the terminal, without need to change anything inside the file.

Create a section named **Appendix** in your report and include all your codes with a text based format (no screenshots). Please don't forget the `Makefile`.

5 Submission On Avenue to Learn

Submit on Avenue:

1. `main.c`
2. `vaderSentiment.c`
3. `utility.h`

4. `README.pdf`
5. `Makefile`

Please do **NOT** submit any zip files.

6 Appendix

Only if you are interested to understand what is the array for scores distribution you should study this section.

The array of integers following each word in the `vader_lexicon.txt` file represents **the distribution of human sentiment ratings** for that word. Each element in the array corresponds to a sentiment score assigned by individual human raters on a scale that usually ranges from -4 to +4, indicating how people generally perceive the emotional valence (positivity or negativity) of the word.

Here's how VADER uses this array of integers: '

1. **Capturing Variability in Sentiment Perception**:
   - The array shows that different people may perceive the sentiment of a word differently. For example, the array [2, 2, 1, 2, 1, 3, 2, 0, 2, 2] for "smart" indicates that while some people rated "smart" with a moderate positive sentiment (2), others rated it even more positively (3), while one person gave a neutral rating (0).
   - This variability helps VADER capture the nuanced sentiment of words that might not have a single, universally agreed-upon sentiment value.
2. **Calculating Mean and Standard Deviation**:
   - VADER uses these ratings to calculate an average score (mean) and a measure of spread (standard deviation) for each word. The **mean** becomes the primary sentiment score (e.g., 1.7 for "smart"), while the **standard deviation** (e.g., 0.78102 for "smart") helps indicate how consistently the sentiment is perceived.
   - A high standard deviation implies more disagreement among raters, while a low standard deviation suggests that most raters agreed on the sentiment.
3. **Enhancing Context Sensitivity**:
   - VADER considers both the mean and standard deviation when interpreting words within phrases or sentences. Words with higher standard deviations might be treated

with more flexibility or adjusted slightly in specific contexts to account for less consensus in sentiment interpretation.

In essence, the array provides a deeper understanding of how varied sentiment perceptions are for each word, helping VADER produce more nuanced and context-aware sentiment scores.