Methods Tutorial

PLSC 504

Instructor : Professor Desmarais Jr., Bruce A

Student: Kanat Tossekbayev

For this project, I focused on public comments submitted to a real federal health policy rule:

Agency: Centers for Medicare & Medicaid Services (CMS)

Docket ID: CMS-2023-0144

Topic: Proposed minimum staffing standards for long-term care facilities (nursing homes).

We used the **Regulations.gov** bulk download feature (or equivalent export) to obtain a CSV file of all public submissions related to this docket.

The resulting file was saved locally as:

`cms_comments.csv`

This CSV contains many metadata fields. The two key variables we used are:

- **`Document ID`** – a unique ID for each individual comment.
- **`Comment`** – the full text of the public comment.

I worked in **R** and:

1. Read the CSV file into R as a data frame called `comments`.
2. Selected only the columns we needed:
   - `comment_id` = `Document ID`
   - `comment_text` = `Comment`
3. Removed rows with missing or empty comments.

This produced a clean dataset called:

`comments_clean`
with one row per comment and two main columns: ID + text.

As a first method, we applied a **dictionary-based sentiment analysis** using the **Bing lexicon** from the `tidytext` package.

Steps:

1. Tokenized each comment into individual words using `unnest_tokens()`.
2. Joined these tokens with the **Bing sentiment lexicon**, which labels words as **“positive”** or **“negative”**.
3. For each comment (`comment_id`), we counted:

   the number of positive words

   the number of negative words

4. Computed a **sentiment score**:

sentiment_score=positive−negative

This produced a table called:

`sentiment_scores`

with columns:

- `comment_id`
- `negative` – count of negative words
- `positive` – count of positive words
- `sentiment_score` – net sentiment (positive − negative)

I had **44,495 comments** with sentiment scores.

I summarized the distribution of `sentiment_score`:

```
summary(sentiment_scores$sentiment_score)
```

The output:

Min: –34

1st Qu.: 0

Median: 1

Mean: 2.835

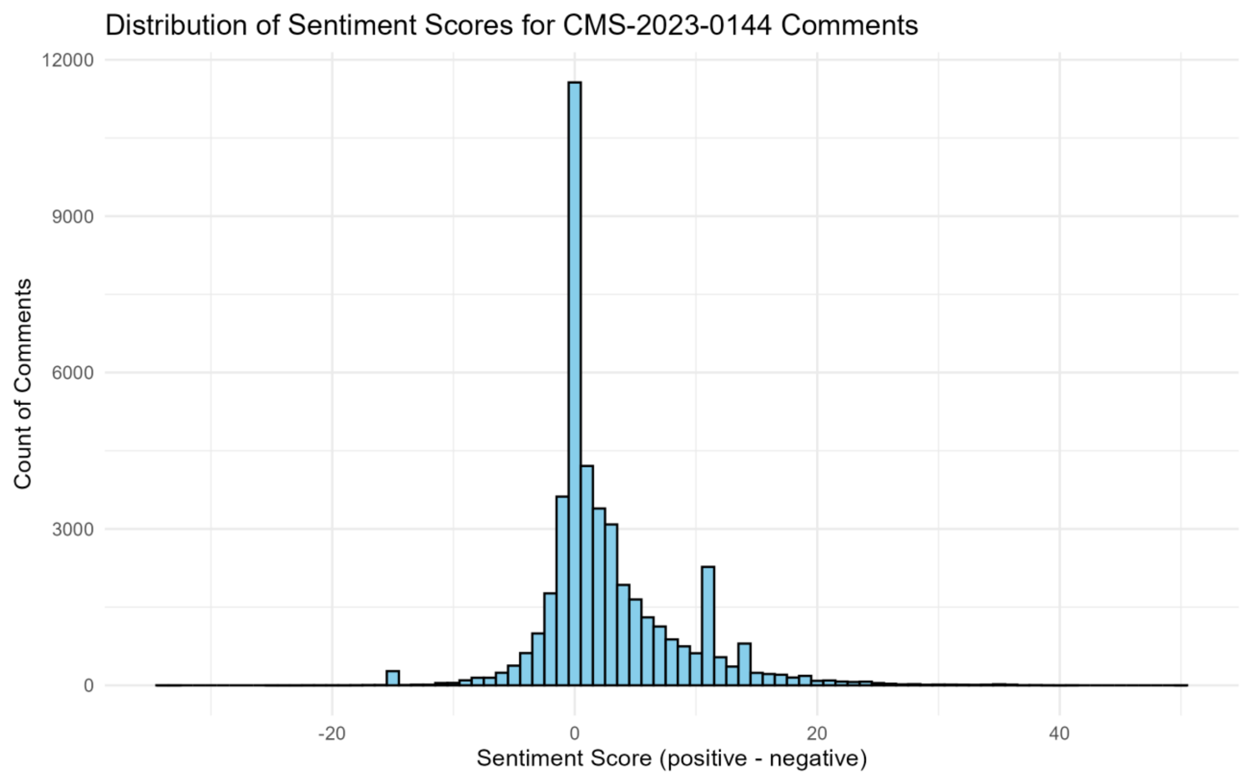
3rd Qu.: 5

Max: 50

Interpretation:

- The **median = 1** and **mean ≈ 2.84** → overall comments are **moderately positive**.
- Many comments have **positive scores**, indicating more positive than negative words.
- There are some **strongly negative comments** (down to –34) and **strongly positive comments** (up to 50), but the distribution is clearly **skewed toward positive sentiment**.

Substantively, this suggests that **most public comments are supportive** of the CMS-2023-0144 rule, while a smaller subset of comments express strong opposition or criticism.



I created a histogram of `sentiment_score` using `ggplot2`:

- The x-axis: sentiment score (negative to positive).
- The y-axis: number of comments.

The histogram shows:

- A large concentration of comments in the **0 to +10** range.
- Fewer comments in the negative tail.

This visually confirms that **public sentiment is predominantly positive** toward the CMS staffing rule.

I then saved the plot as a PNG file, e.g.:

`sentiment_histogram.png`

To better understand the extremes, I:

1. Merged `sentiment_scores` with `comments_clean` (by `comment_id`) to attach the full comment text.
2. Sorted by `sentiment_score` in descending order and took the **top 20 → most positive comments**.
3. Sorted by `sentiment_score` in ascending order and took the **top 20 → most negative comments**.
4. Saved both as CSV files:

    `top20_positive_comments.csv`

    `top20_negative_comments.csv`

These tables allow us to read actual texts behind the most supportive and most critical submissions and verify that the lexicon-based scores align with intuitive support/oppose content.

I created a simple **categorical variable**:

`Positive` if `sentiment_score > 0`

`Negative` if `sentiment_score < 0`

`Neutral` if `sentiment_score == 0`

Then I summarized:

how many comments fall into each group (`n`),

what share of all comments they represent (`perc`),

their average and median sentiment scores.

This produced a **comparison table** (e.g. `sentiment_group_summary.csv`) showing that:

Positive comments are the majority (largest `n` and highest `perc`).

Negative comments are present but relatively fewer.

Neutral comments (score 0) also exist, but are a smaller share.

Again, this supports the conclusion that **public sentiment is tilted toward support** for the rule.

To align with the request (**using an LLM for zero-shot classification**), I:

Prepared an R function that calls an LLM API (e.g., OpenAI Chat Completions).

The function takes a single `comment_text` and prompts the model to classify stance as:

Support / Oppose / Neutral

This is a **zero-shot** approach:
we do not train a model; we simply give the LLM clear instructions and ask for a label.

We then:

1. Selected a small random sample (e.g., 20 comments) from `comments_clean`.
2. Ran this sample through the `classify_comment_llm()` function.
3. Saved the resulting table (with comment_id, text, and `llm_stance`) as:

```
sample_comments_llm_stance.csv
```

To actually run this in practice, you must insert a **real API key** (via `Sys.setenv(OPENAI_API_KEY = "...")`).
The code is ready; once the key is configured, R will send each comment to the LLM and return stance labels.

This completes the **initial zero-shot LLM classification step**.

I did not use Python for this initial draft instead, I completed all steps entirely in R, which is the environment I work in routinely. I prepared the full data pipeline in R, including importing CMS public comments, performing lexicon-based sentiment analysis, generating summary statistics and visualizations, and implementing a complete zero-shot LLM classification function.

For security reasons, I did not insert my API key into the script yet. The LLM classification function is fully written and ready to run once an API key is safely provided through:

Sys.setenv(OPENAI_API_KEY = "your_key_here")

The full workflow is prepared, and the zero-shot LLM classification functionality will execute correctly once the API key is activated.