Project Report: Sentiment Analysis Pipeline

1 Methodology & Steps

For this project, I built a custom Aspect-Based Sentiment Analysis (ABSA) system to find out what users think about specific features of the headphones. The process involved four main steps:

Step 1: Cleaning the Data

First, I cleaned up the raw review text. Since the data had a mix of English and Hindi words, I converted everything to lowercase to make it consistent. I used a regular expression to remove numbers and special symbols but kept the spaces so I could still tell where words started and ended.

Step 2: Finding Keywords (Manual RAKE)

Instead of importing a library, I wrote the **RAKE (Rapid Automatic Keyword Extraction)** algorithm from scratch.

- **How it works:** I split the text into phrases using standard English stopwords (like ”is”, ”the”, ”and”) as dividers.
- **Scoring:** I calculated a score for every phrase based on how often its words appeared together (”Word Degree”) compared to how often they appeared alone (”Word Frequency”).
- **Result:** This helped me find multi-word topics like ”superb bass effort” instead of just single words.

Step 3: Scoring Sentiment

To decide if a review was positive or negative, I used the **SentiWordNet** database.

- **Constraint:** I only looked at **Adjectives**. I filtered out nouns and verbs because they usually just describe the product features without giving an opinion.
- **Context:** For every keyword found in Step 2, I looked at the 3 words before and after it to find these adjectives.

Step 4: Handling Negations

I added a logic check to handle words like ”not” or ”never.” If one of these words appeared near an adjective (e.g., ”not good”), I multiplied the sentiment score by -1. This ensures that ”not good” counts as negative, not positive.

2 Design Decisions

Why build RAKE from scratch?

Using a pre-made library often hides how the scoring works. By coding it myself, I could control exactly how phrases were split. This was important because the reviews had some Hindi text, and standard libraries might have broken those phrases incorrectly. My version kept the longer, more meaningful phrases intact.

Why filter for Adjectives only?

I found that including verbs or nouns introduced too much noise. For example, the word ”headphone” (noun) has a neutral score, but it doesn’t tell us if the user likes it. Adjectives like ”poor”, ”amazing”, or ”broken” are the strongest indicators of actual sentiment, so focusing on them made the final scores more accurate.

Why Aspect-First?

I extracted the aspects *before* calculating sentiment. This is better than just averaging the whole sentence because a single review might praise the ”bass” but criticize the ”wire.” My approach scores these two things separately.